



UNIVERSIDAD NACIONAL
AUTÓNOMA DE
MÉXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA MECÁNICA E INDUSTRIAL

TESIS:

**Desarrollo de un sistema de detección y
manipulación de objetos para un robot de
Servicio**

Edgar de Jesús Vázquez Silva

Noviembre 2016

Índice general

1. Introducción	5
1.1. Planteamiento del problema	5
1.2. Hipótesis	7
1.3. Objetivos	7
2. Marco teórico	9
2.1. Robots de servicio	9
2.2. Fundamentos básicos de Robots Manipuladores	11
2.2.1. Configuraciones típicas y parámetros característicos . .	12
2.2.2. Descripciones espaciales y transformaciones	14
2.2.3. Cinemática directa	15
2.3. Imágenes RGB-D	16
2.4. Algoritmo RANSAC	18
2.4.1. Algoritmo	18
2.4.2. Modelo matemático de un plano	19
2.5. Forma punto-normal y forma general de la ecuación de un plano	21
2.5.1. Modelo del plano definido por tres puntos	22
2.6. Algoritmo PCA	23
2.6.1. Desviación estándar	23
2.6.2. Varianza	24
2.6.3. Covarianza	25
2.6.4. Matriz de covarianzas	26
2.6.5. Eigenvalores y Eigenvectores	26
2.7. Características del objeto	28
2.8. Planeación de tareas	28
3. Detección de objetos	29
3.1. Segmentación de planos con RANSAC	29
3.2. Extracción de objetos	32
3.3. Aproximación de orientación de objetos con PCA	33
3.3.1. Adquisición de datos y cálculo de la media	33

3.3.2. Cálculo de la matriz de covarianza	35
4. Manipulador	37
4.1. Documentación servos dynamixel	37
4.1.1. Motores dynamixel Serie MX	38
4.1.2. Características MX-106	41
4.1.3. Características MX-64	42
4.1.4. Características MX-28	43
4.1.5. Características de conexión	46
4.2. Descripción de los elementos del sistema de manipulación	46
4.3. Cinemática directa	50
4.3.1. Teoría trasformaciones y medidas del brazo robotico .	50
4.3.2. Implementación (Descripción con ROS)	52
4.4. Cinemática inversa	56
5. Integración	57
5.1. Máquinas de estados	57
5.2. ROS	57
5.3. Constitución del robot de servicio Justina	57
6. Resultados y conclusiones	58

Agradecimientos

A mi familia,
al laboratorio de Biorobótica,
al Dr. Jesús Savage Carmona,
al Mtro. Marco Negrete Villanueva,
por su paciencia y apoyo en todo momento.

Se agradece al CONACYT, a través del proyecto 245491, "Laboratorio de Movilidad e Infraestructura Verde para la Eficiencia Energética en Ciudades", por el apoyo recibido en la realización de este documento.

Abstract

El reconocimiento de objetos y la adecuada manipulación de los mismos es una problemática común en el área de la robótica de servicios. El presente documento aborda el diseño de un sistema de manipulación de objetos formado por un brazo robótico de 7DOF, el desarrollo de un algoritmo de visión computacional para reconocer la posición y orientación de los objetos, el desarrollo de la cinemática inversa del brazo robótico y finalmente la planeación de acciones entre la detección de un objeto y su correcta manipulación.

Capítulo 1

Introducción

En los últimos años, el área de la robótica y sus múltiples aplicaciones se han expandido a pasos agigantados, tal es el caso de la robótica de servicios. Años atrás la idea de tener un robot capaz de ayudar en las tareas del hogar sólo era concebida gracias a la ciencia ficción; hoy en día es una total realidad. Un robot es un sistema mecánico controlado automáticamente, reprogramable, mutipropósito, con diversos grados de libertad, el cual puede ser fijo o móvil [1]. Actualmente existe un auge en utilizar a los robots como auxiliares en las actividades domésticas, un área llamada: “robótica de servicio”. Sin embargo el área de la robótica de servicios y robots de asistencia comprende un gran rango de problemáticas.

1.1. Planteamiento del problema

Los robots enfrentan problemáticas a la que cualquier humano está sometido día a día: ambientes dinámicos, características de entornos no estandarizados, incertidumbre ante escenarios desconocidos. Dada la naturaleza de esta disciplina científica han surgido diversas líneas de investigación que abarcan estas problemáticas. La mecánica, la electrónica, la informática, la inteligencia artificial y la ingeniería de control, son algunas de las disciplinas involucradas. La robótica se ayuda de este estas disciplinas para resolver problemas particulares, por ejemplo un robot de servicios debe ser capaz: de reconocer y manipular objetos en diferentes ubicaciones y desde diferentes alturas, de tener locomoción en diferentes tipos de superficies, de interactuar con un humano, de distinguir diferentes personas. Por último, pero no menos importante, el funcionamiento seguro de estos sistemas en ambientes dinámicos es un requisito fundamental para su futura aceptación y aplicabilidad.

La creación de estos sistemas autónomos requiere la integración de un gran conjunto de capacidades y tecnologías. Los ejemplos de capacidades incluyen la interacción humano-robot (habla, identificación de personas, seguimiento de personas, entre otros), navegación, planificación de acciones, control de comportamientos, detección y reconocimiento de objetos, manipulación de objetos o seguimiento de objetos. Con respecto a las tecnologías se requerido la aplicación de sensores RGB-D, cámaras estereoscópicas, sensores láser, entre otros.

Con respecto a la inteligencia, los sistemas deben contener métodos de planificación de acciones y comportamientos adaptables. Los procedimientos apropiados deben, por ejemplo, permitir al operador del robot enseñar nuevos comportamientos y entornos vía comandos de voz o gestos. Los futuros hogares probablemente contendrán dispositivos electrónicos más inteligentes capaces de comunicarse entre si incluyendo el uso de internet como base común de conocimientos, de modo que los robots desempeñarán un papel más importante.

Entre todas estas líneas de investigación es imprescindible contar con un robot que sea capaz de interactuar con los objetos en el mundo real, por ello es necesario contar un sistema que pueda reconocer los objetos y su posición adecuadamente. Sin embargo, esta línea de investigación atiende a problemáticas muy concretas, en el mundo real los robots se enfrentan con condiciones dinámicas en el ambiente, por ejemplo al pedir a un robot que tome un objeto y pueda llevárselo hasta nosotros, el primer problema al que nos enfrentamos es conocer la posición del objeto (la cual será diferente en cada ocasión), posteriormente, si deseamos localizar dos objetos del mismo tipo, estos no serán reconocidos de igual manera por el robot debido a las condiciones de luz, a los cambios de forma, y a los cambios de apariencia.

Dadas las características antes descritas, es preciso estimar la probabilidad de que un robot pueda manipular correctamente los objetos. Para ello es necesario dividir la tarea en operaciones: la primera de ellas consiste en estimar la posición del objeto, tener un indicador que nos ayude a determinar cuál es la mejor forma de tomar un objeto y conocer la probabilidad de que el robot haya reconocido un objeto exitosamente. Otra de las operaciones necesarias es llevar el actuador final del brazo robótico a una posición (x , y , z) deseada. Por último es necesario contar con un planeador de acciones para coordinar cada uno de los eventos dentro de la tarea.

Podemos observar que las problemáticas son variadas, sin embargo podemos reducir el problema principal en tres tareas secundarias: Detectar un objeto, obtener una aproximación de cuál será la mejor manera de tomarlo y caracterizar el sistema en conjunto para obtener un parámetro de confiabilidad.

1.2. Hipótesis

- Un sistema de visión computacional con implementación de un algoritmo de análisis de componentes principales podrá indicarnos cuál es la mejor orientación para tomar un objeto y por tanto mejorará manipulación de objetos.

1.3. Objetivos

Objetivo general

Implementar un conjunto de algoritmos de visión computacional que mejore el sistema de detección y manipulación de objetos formado por un sensor kinect y un brazo robótico de 7DOF en un robot de servicios.

Objetivos específicos

- Implementar un algoritmo de visión computacional para identificar un plano.
- Implementar un algoritmo de visión computacional para determinar la posición de un objeto.
- Implementar un algoritmo de análisis de componentes principales para identificar la orientación de los objetos.
- Calcular la cinemática inversa de un brazo robótico de 7DOF para llevarlo a una posición deseada ($x, y, z, \text{roll}, \text{pitch}, \text{yaw}$).
- Caracterizar el sistema de manipulación de objetos usando un modelo bayesiano.

En el capítulo 3 de este documento se aborda la manera en que se desarrollaron los algoritmos de visión computacional para realizar la extracción de un plano, identificar la posición de un objeto, implementar un algoritmo de Análisis de Componentes Principales para obtener cual será la mejor orientación para tomar un objeto a partir de su forma. En el capítulo 4 se describe el sistema de manipulación utilizado. Se comienza por conocer las características de los actuadores utilizados, se obtienen las ecuaciones correspondientes a la cinemática de los brazos para llevarlo hasta una posición (x, y, z, roll, pitch, yaw) deseada. En el capítulo 5 se realiza la integración de las tareas antes mencionadas, la plataforma sobre la cual se desarrolló el sistema y se describe el desarrollo del modelo probabilístico utilizado para mejorar la manipulación de objetos en un robot de servicios.

Capítulo 2

Marco teórico

2.1. Robots de servicio

Dado que este trabajo se centrará en los robots de servicio es importante mencionar que un robots de servicio es un robot capaz de realizar tareas de la vida diaria en un ambiente similar al de un hogar real. Actualmente el desarrollo de este tipo de robot está guiado por la competencia internacional Robocup @Home.[2] Esta categoría promueve la incorporación de habilidades robóticas avanzadas para la interacción con los humanos y con el entorno de operación. Esta competencia se enfoca a desarrollar habilidades en los robots de servicios. Ejemplos de estas habilidades son la localización y navegación segura en ambientes no controlados, la comunicación natural humano-robot por voz y gestos y habilidades visuales para el reconocimiento y la manipulación de objetos.[3]

Los robots de servicio enfrentan diversos retos: desarrollar tareas en ambientes dinámicos, características de entornos no estandarizados, incertidumbre ante escenarios desconocidos. Dadas las condiciones en que estos robots operan los programadores y desarrolladores de robots de servicios han optado por dotar a los robots de ciertas capacidades. Por ejemplo el robot Cosero, consta de un sistema de percepción del entorno, un sistema de manipulación, un sistema de interacción humano-robot y un sistema de planeación de tareas.

Tabla con las características de los robots de servicio.

En particular en una metodología de detección y manipulación de objetos, resulta una tarea fundamental a resolver en los robots de servicio. La idea general de la robótica de servicio doméstico ha existido desde hace mucho



Figura 2.1: Estructura de software del robot Cosero.



Figura 2.2: Estructura de software del robot Cosero.

tiempo, pero es un tema de investigación relativamente joven. El objetivo de crear robots de servicios útiles y autónomos que puedan interactuar con seres humanos y objetos en el mundo real en un entorno natural plantea un gran número de problemas sin resolver en muchas disciplinas científicas.

Recientemente, el progreso en estos campos de investigación, así como el progreso y la normalización en el desarrollo de hardware y software, ha llevado a un aumento en la disponibilidad de recursos, métodos y componentes para el desarrollo de Robots de servicio doméstico . Por ello estamos cada vez más cerca de convivir con robots de servicios de manera exitosa en diversos lugares, por ejemplo hospitales [4], oficinas, construcciones, o tiendas departamentales [5].



Figura 2.3: Estructura de software del robot Cosero.

Estos desarrollos han sido posibles gracias a herramientas de código abierto como lo es Ubuntu, ya que ha servido como base para el desarrollo de software especializado para robots, por ejemplo ROS [6]. Este conjunto de librerías especializadas para robots pueden llegar a ser muy particulares y estar enfocados a una sola área de investigación de las antes mencionadas, por ejemplo: Carmen [7], un conjunto de librerías y algoritmos de control para navegación de robots, desarrollados por la universidad Carnegie Mellon[8].

En la parte de simulación se cuenta con ejemplos como el USARSim [9], rviz [10] o Gazebo [11]. Por lo que respecta a los algoritmos de visión computacional existen bibliotecas de código abierto por ejemplo OpenCV [12] el cual tiene un gran campo de aplicaciones[13]. Por lo que corresponde a los kits estándar de hardware, podemos mencionar la construcción de robots de plataforma estándar por ejemplo VolksBot [14] y las plataformas bases por ejemplo ActivRobots [15], Pepper [16], Asimo [17] ha hecho posible desarrollar software de manera más rápida y eficiente.

2.2. Fundamentos básicos de Robots Manipuladores

La manipulación adecuada de objetos es una característica imprescindible en los robots de servicio dadas las condiciones de su entorno y las tareas cotidianas que le podemos asignar a dicho robot. Una posible solución a la problemática de la manipulación de objetos es la incorporación de manipuladores seriales a una base móvil; sin embargo se debe tener en cuenta que



Figura 2.4: Robot asimo desarrollado por la compañía Honda.

los objetos a ser manipulados se encuentran en condiciones aleatorias de posición y orientación, estas características implican que el manipulador serial debería ser capaz de alcanzar una posición (x , y , z) con cualquier orientación (roll, pitch, yaw).

Los robots manipuladores clásicos presentan una configuración antropomórfica serial, que hace semejanza con un brazo humano. La arquitectura típica de un manipulador consiste en una serie de barras rígidas unidas entre sí mediante el uso de articulaciones rotacionales o prismáticas. De manera general cada articulación logra su movimiento gracias a un actuador y a la adición de algunos elementos complementarios como sensores de posición y de velocidad[18].

Los robots manipuladores clásicos están caracterizados, desde el punto de vista mecánico, por una serie de propiedades tales como los grados de libertad, el espacio de trabajo, la rigidez estructural, peso propio y la capacidad de llegar a un punto deseado con exactitud en múltiples repeticiones. Además en los robots manipuladores suelen tomarse en cuenta otras características adicionales: la carga útil máxima y la velocidad de trabajo.

2.2.1. Configuraciones típicas y parámetros característicos

Según la geometría de su estructura mecánica, un manipulador puede ser:



Figura 2.5: Robot serial elaborado por la compañía ABB.

- Cartesiano, cuyo posicionamiento en el espacio se lleva a cabo mediante articulaciones lineales.
- Cilíndrico, con una articulación rotacional sobre una base y articulaciones lineales para el movimiento en altura y en radio.
- Polar, que cuenta con dos articulaciones rotacionales y una lineal.
- Esférico (o de brazo articulado), con tres articulaciones rotacionales.
- Mixto, que posee varios tipos de articulaciones, combinaciones de las anteriores. Es destacable la configuración SCARA (Selective Compliance Assembly Robot Arm).
- Paralelo, posee brazos con articulaciones prismáticas o rotacionales concurrentes. Figura 2.3

Los principales parámetros que caracterizan a los robots industriales son:

- Número de grados de libertad. Es el número total de grados de libertad de un robot, dado por la suma de g.d.l. de las articulaciones que lo componen. Aunque la mayoría de las aplicaciones industriales requieren 6 g.d.l., como las de soldadura, mecanizado y almacenamiento, otras más complejas requieren un número mayor, tal es el caso de las labores de montaje.
- Espacio de accesibilidad o espacio (volumen) de trabajo. Es el conjunto de puntos del espacio accesibles al punto terminal, que depende de la configuración geométrica del manipulador. Un punto del espacio se dice totalmente accesible si el efecto final puede situarse en él en todas las orientaciones que permita la constitución del manipulador y se dice

parcialmente accesible si es accesible por el efecto final pero no en todas las orientaciones posibles. En la figura inferior se aprecia el volumen de trabajo de robots de distintas configuraciones.

- Capacidad de posicionamiento del punto terminal. Se concreta en tres magnitudes fundamentales: resolución espacial, precisión y repetibilidad, que miden el grado de exactitud en la realización de los movimientos de un manipulador al realizar una tarea programada.
- Capacidad de carga. Es el peso que puede transportar el elemento terminal del manipulador. Es una de las características que más se tienen en cuenta en la selección de un robot dependiendo de la tarea a la que se destine.
- Velocidad. Es la máxima velocidad que alcanzan el efecto final y las articulaciones.



Figura 2.6: Robot paralelo.

2.2.2. Descripciones espaciales y transformaciones

La cinemática es la ciencia del movimiento que trata el tema sin considerar las fuerzas que lo ocasionan. Dentro de esta ciencia se estudian la posición, la velocidad y la aceleración. En consecuencia, el estudio de la cinemática de manipuladores se refiere a todas las propiedades geométricas y las basadas en los cambios de estas a lo largo del tiempo. Dadas las características de este trabajo solo se abordarán la cinemática directa e inversa, sin llegar a analizar la dinámica del manipulador.



Figura 2.7: Esquema de sistemas de referencia en robot antropomórfico de seis grados de libertad.

El problema de la cinemática directa se plantea en términos de encontrar una matriz de transformación que relaciona el sistema de coordenadas ligado al cuerpo en movimiento respecto a un sistema de coordenadas que permanece estático y se toma como referencia. Para lograr esta representación se usa la matriz de transformación homogénea con una dimensión 4x4, la cual incluye las operaciones de rotación y translación.

La matriz de transformación homogénea es una matriz de 4x4 que transforma un vector expresado en coordenadas homogéneas desde un sistema de coordenadas hasta otro sistema de coordenadas. La matriz de transformación homogénea tiene la siguiente estructura:

$$T_{0,1} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

donde los vectores n , s , a , son vectores ortogonales unitarios que representan la rotación del sistema y p es un vector que describe la posición x , y , z del origen del sistema actual respecto del sistema de referencia.

2.2.3. Cinemática directa

El problema de la cinemática directa consiste en determinar la posición del efecto final en el espacio dado el valor de cada una de las articulaciones. El valor de los ángulos de las articulaciones se determinan con ayuda

de sistemas de referencia ubicados en cada una de las articulaciones del robot.

Un robot manipulador está compuesto de un conjunto de enlaces conectados por varias juntas. Las articulaciones pueden ser muy simples, tales como una articulación de revolución o una articulación prismática. Una articulación de revolución es como una bisagra y permite una rotación relativa alrededor de un solo eje coordenado; una junta prismática permite un movimiento lineal a lo largo de un solo eje. En ambos casos podemos observar el que la articulación tiene un solo grado de libertad de movimiento: el ángulo de rotación en el caso de una articulación de revolución, y la cantidad de desplazamiento lineal en el caso de una articulación prismática.

Con el supuesto de que cada articulación tiene un solo grado de libertad, la acción de cada una de las articulaciones se puede describir por un solo número real: el ángulo de rotación en el caso de una articulación de revolución o el desplazamiento en el caso de una junta prismática. El objetivo del análisis de la cinemática directa es determinar el efecto acumulativo de todo el conjunto de variables de la articulación y observarlo en el efecto final.[19]

El análisis cinemático de un manipulador de n -enlaces puede ser extremadamente complejo y las convenciones que se presentan a continuación simplifican el análisis considerablemente.

Un robot manipulador con n grados de libertad tendrá por consecuencia n articulaciones. Se numeran las articulaciones de 1 a n , y se numeran los enlaces de 0 a n , comenzando desde la base en un sistema de referencia fijo. Para esta convención, la articulación i conecta el enlace $i - 1$ al enlace i .

Cuando se acciona la articulación i el enlace se mueve, por lo tanto, el enlace 0 (el primer eslabón) es fijo, y no se mueve cuando las juntas consecuentes son accionadas. Por supuesto, el robot manipulador podría ser móvil (por ejemplo, podría ser montado en una plataforma móvil o en un vehículo autónomo), pero para esta primera parte abordaremos el problema suponiendo una plataforma fija.

2.3. Imágenes RGB-D

En la robótica de servicios resulta imprescindible contar con robots que sean capaces de percibir su entorno, los elementos que los rodean y deter-

minar características de los mismos. Los humanos, dadas estas necesidades, hemos desarrollado el sentido de la vista que nos permite determinar características del entorno y de los objetos que lo componen tales como el color, la forma, las dimensiones ó su ubicación en el espacio.

Para optimizar los movimientos de un robot, no sólo se debe identificar cada objeto que se encuentra en el entorno de trabajo, sino también la posición que estos guardan respecto al robot. Normalmente, la segmentación de objetos de una imagen se logra mediante la segmentación de color, esta segmentación se logra a partir del análisis cromático de los componentes RGB de la imagen obtenida del objeto; sin embargo, este método es poco eficiente debido a que es sumamente susceptible a los cambios en la iluminación. [20]

Por lo tanto, necesitamos considerar una fuente de datos adicional, en este caso la profundidad, para discriminar objetos que no están en el rango de interés. En este documento se reportan los datos obtenidos al resolver este problema con la incorporación de un sensor RGB-D.

La rápida evolución de esta reciente tecnología ha resultado en el aumento de la calidad de las imágenes, la mejora en la sincronización del color con la profundidad y la disminución del tiempo de muestreo entre cada imagen. En particular, la profundidad de la imagen se obtiene en una matriz resultado de la proyección de rayos infrarrojos sobre los objetos. El sensor cuenta con una cámara infrarroja que obtiene la lectura de los haces infrarrojos reflejados sobre los objetos.

El sensor Kinect incorpora un sensor de profundidad, un cámara de color y un arreglo de cuatro micrófonos que, en conjunto, proporcionan una captura de movimiento de cuerpo completo en 3D. La figura 2.5 muestra la disposición del proyector de infrarrojos (IR), la cámara de color y la cámara IR. El sensor de profundidad consta de un proyector IR combinado con una cámara IR, la cual es un sensor semiconductor de metal-óxido monocromático. El sensor de profundidad es desarrollado por la compañía israelí PrimeSense.

Aunque la tecnología exacta es cerrada, se basa en el principio de la estructura de la luz y su difracción. El proyector IR es un láser IR que pasa a través de una rejilla de difracción y se esparce como una nube de puntos. Dada la geometría de construcción entre el proyector IR y la cámara IR se puede reconstruir un modelo 3D si se observa un punto en el proyector y posteriormente se ubica su correspondiente reflexión en la cámara, esto se



(a) Sensor kinect

(b) Sensor kinect estructura

Figura 2.8: Sensor kinect elaborado por Microsoft.

puede lograr utilizando triangulación.

Debido a que el patrón de puntos es relativamente al azar, la coincidencia entre la imagen IR y el patrón del proyector puede realizarse de forma directa comparando pequeñas vecindades utilizando, por ejemplo, referencias cruzadas.[21]

El valor de la profundidad se codifica en escala de grises, cuanto más oscuro sea un píxel, más cerca está el punto de la cámara en el espacio, los píxeles negros indican que no se posee información respecto a la distancia en ese píxel. Esto puede suceder si los píxeles están demasiado lejos o demasiado cerca de la cámara IR.

2.4. Algoritmo RANSAC

2.4.1. Algoritmo

El algoritmo RANSAC (RANdom Sample And Consensus) fue introducido por primera vez por Fischler y Bolles en 1981, como un método para estimar los parámetros de un determinado modelo a partir de un conjunto de datos contaminados por grandes cantidades de outliers. Un dato es considerado un outlier si no encaja en el modelo propuesto, dentro de un umbral de error que define el valor máximo desviación atribuible al efecto del ruido.

RANSAC selecciona uniformemente al azar un subconjunto de muestras de datos y lo utiliza para crear un modelo de estimación de parámetros. Luego determina si las muestras que están dentro de una tolerancia de error del modelo antes propuesto. Si estas muestras se consideran como parte del modelo generado se agrupan en un conjunto llamado *consenso*: De manera

análoga se llama outliers el conjunto de los que superan el umbral de error con respecto al modelo. Si el recuento de los datos agrupados en el consenso es suficientemente alto con respecto de los outliers, se considera que el modelo se ajusta al conjunto de datos, siendo el modelo propuesto un modelo que se aproxima al conjunto total de datos.

A continuación se muestra el pseudocódigo básico para el algoritmo RANSAC.

```

Parámetros:
datos – conjunto de datos observados
modelo – una propuesta de modelo que puede ajustar al conjunto de datos
n – el número mínimo de datos con los cuales se considera un buen modelo
k – el número máximo de iteraciones para algoritmo
t – el valor del umbral que determinará si un punto se justa al modelo

Salidas:
mejorAjuste – modelo encontrado con los parámetros o una variable vacía si no se encontró ningún
modelo que justara.

Inicialización de variables:
iteraciones = 0
mejorModelo = vacío
mejorError = Algo realmente grande

while iteraciones < k do
    Generación del modelo a partir de datos aleatoriamente seleccionados.
    modeloPropuesto = f(datos)
    inliers = 0
    for cada punto in datos and not en inliers do
        calcular error
        if punto ajusta en modeloPropuesto with an error < than t then
            añadir punto a inliers
            errorModelo = errorModelo + error
        end if
    end for

    if número de elementos in inliers is > n then
        esto significa que tenemos un buen modelo
        Verificamos si el modelo calculado supera al mejor modelo calculado con anterioridad.
        if errorModelo < mejorError then
            mejorAjuste = mejorModelo
            mejorError = errorModelo
        end if
    end if
    incrementamos iteraciones
end while
Llegamos al numero de iteraciones k y devolvemos el mejor modelo obtenido.

return mejorAjuste

```

2.4.2. Modelo matemático de un plano

En matemáticas, un plano es una superficie plana, bidimensional que se extiende infinitamente lejos. Un plano es el análogo bidimensional de un pun-

to (dimensiones cero), una línea (una dimensión) y un espacio tridimensional. Los planos pueden surgir como subespacios de algún espacio de dimensión superior, como si las paredes de una habitación estuviesen extendidas infinitamente, o pudieran disfrutar de una existencia independiente por derecho propio, como en el caso de la geometría euclíadiana.

Cuando se trabaja exclusivamente en el espacio euclíadiano bidimensional, se utiliza el artículo definido, por lo que el plano se refiere a todo el espacio. Muchas tareas fundamentales en matemáticas, geometría, trigonometría, teoría de gráficas y representación gráfica se realizan en un espacio bidimensional, o, en otras palabras, en el plano.

En un espacio euclíadiano de cualquier número de dimensiones, un plano está determinado únicamente por cualquiera de los siguientes:

- Tres puntos no colineales (puntos no en una sola línea).
- Una línea y un punto no en esa línea.
- Dos líneas distintas, pero que se cruzan.
- Dos líneas paralelas.

Las siguientes afirmaciones se mantienen en el espacio euclíadiano tridimensional, pero no en las dimensiones superiores, aunque tienen análogos de mayor dimensión:

- Dos planos distintos son paralelos o se intersecan en una línea.
- Una línea es paralela a un plano, lo cruza en un solo punto, o está contenida en el plano.
- Dos líneas distintas perpendiculares al mismo plano deben ser paralelas entre sí.
- Dos planos distintos perpendiculares a la misma línea deben ser paralelos entre sí.

2.5. Forma punto-normal y forma general de la ecuación de un plano

De manera análoga a la forma en que se describen las líneas en un espacio bidimensional utilizando una forma de pendiente de puntos para sus ecuaciones, los planos en un espacio tridimensional tienen una descripción natural que utiliza un punto en el plano y un vector ortogonal a él (Vector normal) para indicar su inclinación:

Específicamente, sea r_0 el vector de posición de algún punto $P_0 = (x_0, y_0, z_0)$, y sea $n = (a, b, c)$ un vector no nulo. El plano determinado por el punto P_0 y el vector n consiste en aquellos puntos P , con el vector de posición r , de tal manera que el vector trazado de P_0 a P sea perpendicular a n . Recordando que dos vectores son perpendiculares si y sólo si su producto punto es cero, se deduce que el plano deseado puede ser descrito como el conjunto de todos los puntos r tales que:

$$n \cdot (r - r_0) = 0 \quad (2.1)$$

(El punto aquí significa un producto punto, no la multiplicación escalar.) Ampliado esto se convierte

$$A(x - x_0) + B(y - y_0) + C(z - z_0) = 0 \quad (2.2)$$

Que es la forma punto-normal de la ecuación de un plano. Esta es sólo una ecuación lineal

$$ax + by + cz + d = 0 \quad (2.3)$$

donde:

$$d = -(ax_0 + by_0 + cz_0) \quad (2.4)$$

Es un plano que tiene como vector el vector $n = (a, b, c)$. Esta ecuación familiar para un plano se llama la forma general de la ecuación del plano.

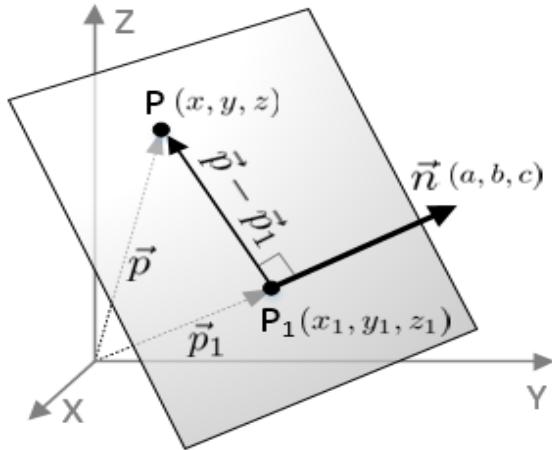


Figura 2.9: Ecuación de un plano definido por un punto y su normal.

2.5.1. Modelo del plano definido por tres puntos

Sea $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_2, y_2, z_2)$, y $p_3 = (x_3, y_3, z_3)$ puntos en el espacio R^3 no-colineales.

Este plano también puede ser descrito por el "punto y un vector normal" prescripción anterior. Un vector normal adecuado se da por el producto cruz.

$$n = (p_2 - p_1) \times (p_3 - p_1) \quad (2.5)$$

Y el punto r_0 puede considerarse como cualquiera de los puntos dados p_1 , p_2 o p_3 (o cualquier otro punto en el plano).

Distancia desde un punto a un plano.

Para un plano $\Pi : ax + by + cz + d = 0$ y un punto $P_1 = (x_1, y_1, z_1)$ no necesariamente en el plano, la distancia más corta de P_1 al plano es

$$D = \frac{|ax_1 + by_1 + cz_1 + d|}{\sqrt{a^2 + b^2 + c^2}} \quad (2.6)$$

De lo cual podemos observar que el punto p_1 se encuentra en el plano si

y sólo si $D = 0$.

Si $\sqrt{a^2 + b^2 + c^2} = 1$ significa que a, b, y c están normalizados, entonces la ecuación se convierte en:

$$D = |ax_1 + by_1 + cz_1 + d| \quad (2.7)$$

2.6. Algoritmo PCA

El Análisis de Componentes Principales (PCA) ha sido llamado Uno de los resultados más valiosos de la álgebra lineal. PCA Se usa abundantemente en muchas formas de análisis, desde la neurociencia a la computación gráfica debido a que es un método simple. El PCA proporciona una metodología para reducir un conjunto de datos complejos a una dimensión inferior.

El algoritmo PCA tiene sus bases en las estadísticas y parte de la idea de que se cuenta con un gran conjunto de datos, y desea analizar ese conjunto en términos de las relaciones entre los puntos individuales dentro de ese conjunto de datos. Se pretende analizar algunas de las medidas que podemos calcular en un conjunto de datos, y lo que estas medidas dicen acerca de los datos en sí.

El tema entero de las estadísticas se basa en la idea de que usted tiene este gran conjunto de datos, Y desea analizar ese conjunto en términos de las relaciones entre un puntos de ese conjunto y el resto de los datos. A continuación se intenta explicar las matemáticas detrás de PCA.

2.6.1. Desviación estándar

La desviación estándar es una medida de la dispersión de un conjunto de datos respecto de la media. Se calcula como la raíz cuadrada de la varianza, determinando la variación entre cada punto perteneciente al conjunto de datos respecto a la media. Si los puntos de datos están más lejos de la media, hay una desviación más alta dentro del conjunto de datos.

Por tanto para calcular la desviación estándar primero debemos calcular la media del conjunto de datos.

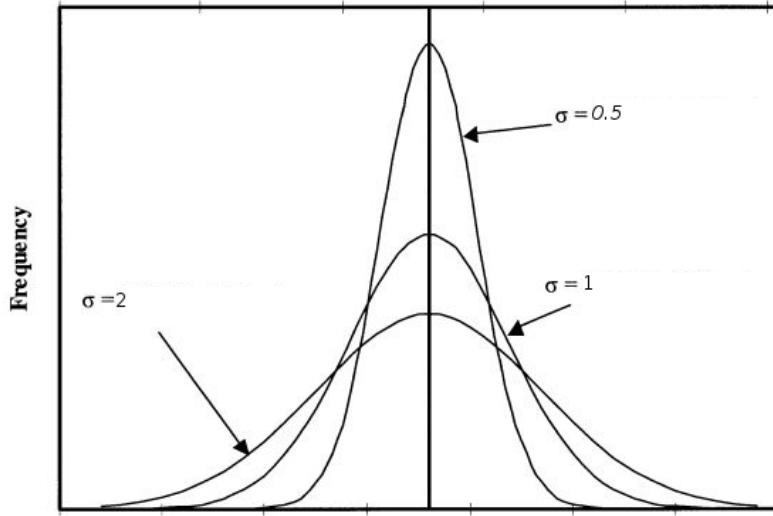


Figura 2.10: Distribución normal con media cero y diferentes desviaciones estándar.

Cálculo de la media de un conjunto de datos.

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n} \quad (2.8)$$

Donde n es el numero total de datos contenido en la muestra.

Cálculo de la desviación estándar.

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n - 1)}} \quad (2.9)$$

2.6.2. Varianza

La noción de varianza se suele emplear en el ámbito de la estadística. Es utilizada para identificar a la media de las desviaciones cuadráticas de una variable de carácter aleatorio, considerando el valor medio de ésta.

La varianza de las variables aleatorias, por lo tanto, consiste en una medida vinculada a su dispersión. Se trata del valor esperado del cuadrado de

la desviación de esa variable considerada frente su media y se mide en una unidad diferente. Por ejemplo: en los casos en que la variable mide una distancia en kilómetros, su varianza se expresa en kilómetros al cuadrado.

Cabe destacar que las medidas de dispersión (también identificadas con el nombre de medidas de variabilidad) se encargan de expresar la variabilidad de una distribución por medio de un número, en los casos en que las diferentes puntuaciones de la variable están muy alejadas de la media. A mayor valor de la medida de dispersión, mayor variabilidad. En cambio, a menor valor, más homogeneidad.

Por lo que respecta a su representación matemática se puede expresar como el cuadrado del valor de la desviación estándar, como se muestra a continuación.

$$s = \sigma^2 \quad (2.10)$$

$$s = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n - 1)} \quad (2.11)$$

2.6.3. Covarianza

Las dos últimas medidas que hemos observado son puramente uni-dimensionales. Conjuntos de datos como este podrían ser: alturas de todas las personas en la sala, las calificaciones para el último examen, etc. Sin embargo, muchos conjuntos de datos tienen más de una dimensión y el objetivo del Análisis de Componentes Principales son estos conjuntos de datos. En tal caso es importante observar si existe alguna relación entre las diferentes dimensiones. Por ejemplo, podríamos tener como nuestro conjunto de datos tanto la altura de todos los estudiantes en una clase, y la calificación que recibieron en un examen.

Podríamos entonces realizar un análisis estadístico para ver si la altura de un estudiante tiene algún efecto en su calificación. La desviación estándar y la varianza sólo funcionan en una dimensión, de modo que podríamos solamente calcular la desviación estándar para cada dimensión del conjunto de datos de forma independiente de las otras dimensiones. Sin embargo, es útil tener una medida similar para mucho las dimensiones varían de la media con

respecto a la otra. La covarianza es tal medida. Covarianza siempre se mide entre 2 dimensiones. Si se calcula la covarianza entre una dimensión y sí, se obtiene la diferencia.

Por lo tanto, si se tiene un conjunto de datos en 3 dimensiones (x, y, z) , entonces se podría medir la covarianza entre: las dimensiones (x, y) , las dimensiones (x, z) y las dimensiones (y, z) . Como se puede observar la medición de la covarianza entre (x, x) , (y, y) o (z, z) se puede reducir a calcular la varianza de cada dimensión respectivamente. La fórmula para la covarianza es muy similar a la fórmula para la varianza. La formula:

$$var(X) = \frac{\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})}{(n - 1)} \quad (2.12)$$

$$cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n - 1)} \quad (2.13)$$

2.6.4. Matriz de covarianzas

Cuando se dispone de un conjunto de datos multidimensionales la información obtenida de los cálculos de las covarianzas se dispone en forma de matriz para facilitar su manejo, tal matriz es llamada Matriz de covarianzas: La matriz de covarianzas es de dimensiones $n \times n$ donde n es el número de dimensiones totales en el conjunto de datos.

A continuación se muestra la matriz de covarianzas para un conjunto de datos de 3 dimensiones, llamadas x, y , y z .

$$C = \begin{bmatrix} cov(x, x) & cov(x, y) & cov(x, z) \\ cov(y, x) & cov(y, y) & cov(y, z) \\ cov(z, x) & cov(z, y) & cov(z, z) \end{bmatrix} \quad (2.14)$$

2.6.5. Eigenvalores y Eigenvectores

Se conocen como valores propios de una matriz al conjunto de valores que resuelven la ecuación:

$$|A - \lambda I| = 0 \quad (2.15)$$

$$\det(A - \lambda I) = 0 \quad (2.16)$$

También puede observarse que λ es un vector de valores escalares que a su vez son solución del polinomio de grado n . Donde n es la dimensión de la matriz A :

Polinomio característico.

$$|A - \lambda I| = (\lambda_1 - \lambda)(\lambda_2 - \lambda)(\lambda_3 - \lambda)\dots(\lambda_n - \lambda) \quad (2.17)$$

Por lo tanto, existe un eigenvector asociado a cada uno de los valores propios de la matriz A

$$(A - \lambda I)v = 0 \quad (2.18)$$

Para encontrar el valor de los vectores característicos de la matriz A se determinan los valores característicos de la matriz A se resuelve la ecuación:

$$\begin{aligned} (A - \lambda_1 I)\vec{v}_1 &= 0 \\ (A - \lambda_2 I)\vec{v}_2 &= 0 \\ &\cdot \\ &\cdot \\ (A - \lambda_3 I)\vec{v}_n &= 0 \end{aligned} \quad (2.19)$$

Es importante observar que el conjunto de todos los vectores propios asociados a un valor propio dado \vec{E}_λ , junto con el vector 0 , forman un espacio vectorial.

Por lo tanto, \vec{E}_λ representa un subespacio de R^n porque es el espacio solución de un sistema homogéneo de ecuaciones lineales y está conformado por matrices columna \vec{v} de $nx1$ que pertenecen a R^n .

Los vectores característicos $\vec{v}_1, \vec{v}_2, \vec{v}_3\dots\vec{v}_n$, asociados a los valores característicos distintos $\lambda_1, \lambda_2, \dots, \lambda_n$ de la matriz A , **son linealmente independientes, y este conjunto es una base en R^n** . Si la matriz A es simétrica

y real, entonces la base es ortogonal.

Es importante resaltar estas dos características puesto que la aplicación que se desarrollará en este trabajo se basa en el principio de la independencia lineal de los vectores propios y que estos construyen una base ortogonal del conjunto de datos observados.

2.7. Características del objeto

2.8. Planeación de tareas

Capítulo 3

Detección de objetos

En este capítulo se trata el desarrollo de los algoritmos propuestos en la metodología de este trabajo. Se comenzará con los desarrollos realizados en el ámbito de la visión computacional:

- Segmentación de planos
- Extracción de planos
- Segmentación de objetos
- Cálculo del centroide de un objeto
- Aproximación a la orientación de un objeto

Posteriormente se abordarán los temas respectivos la cinemática de un brazo robótico de 7DOF. En este apartado abordaremos la cinemática directa del mismo y como trabajo futuro queda la cinemática inversa del mismo.

Es importante mencionar que los desarrollos reportados en este trabajo fueron realizados en el framework para el desarrollo de software para robots, ROS por sus siglas en inglés. Alojado en el sistema operativo Ubuntu.

3.1. Segmentación de planos con RANSAC

Para el desarrollo del algoritmo RANSAC se utilizó el sensor Kinect dado que este sensor, además de una imagen de color, proporciona información sobre la profundidad de los objetos. Con un manejo adecuado de ambos datos

se pueden conseguir avances notables en el campo de la visión computacional.

Para el desarrollo del algoritmo se hizo uso de la información de profundidad de los objetos.

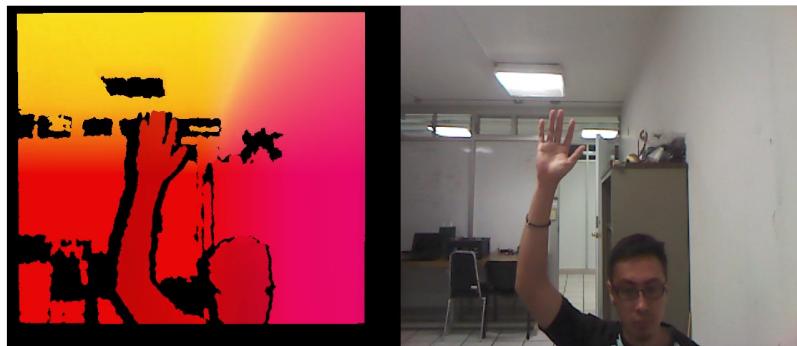


Figura 3.1: Imagen típica de la información otorgada por el sensor Kinect.

Se desarrolló un programa en lenguaje C++ para segmentar los planos. La información de profundidad entregada por el Kinect, se da en una matriz donde cada elemento i, j contiene la información de profundidad obtenida por el sensor, como se muestra a continuación:

Cada elemento de la matriz contiene información de un punto $p = (x, y, z)$ donde el haz infrarrojo fue reflejado, esto nos permite conocer la distancia a la cual se encuentra un objeto, o su superficie. La información que nos entrega el sensor en forma de matriz es de dimensiones 640×480 por tanto consta de 307,200 elementos en total. Dentro de este conjunto de datos procura encontrar las superficies planas.

Se programó el algoritmo RANSAC para extraer planos en C++, el código se puede consultar en la página. ***([github ref](#)).

Para la correcta implementación del algoritmo fue necesario tomar algunas consideraciones, por ejemplo: descartar del procesamiento todos aquellos píxeles que no proporcionen información relevante de profundidad, la construcción de una clase para generar planos.

Para la construcción de planos se tomaron tres puntos aleatorios y se construyeron dos vectores a partir de los puntos y se realizó el producto cruz

entre estos.

$$\begin{aligned} p_1 &= \text{rnd(datos)} \\ p_2 &= \text{rnd(datos)} \\ p_3 &= \text{rnd(datos)} \\ \vec{v}_1 &= p_1 - p_2 \\ \vec{v}_2 &= p_1 - p_3 \\ \vec{N} &= \vec{v}_1 \times \vec{v}_2 \end{aligned}$$

Conocemos la ecuación general del plano:

$$\Pi : Ax + By + Cz + D = 0$$

donde: A, B, C son las componentes del vector normal respectivamente y D la podemos calcular como:

$$D = (A * p_1.x + B * p_1.y + C * p_1.z)$$

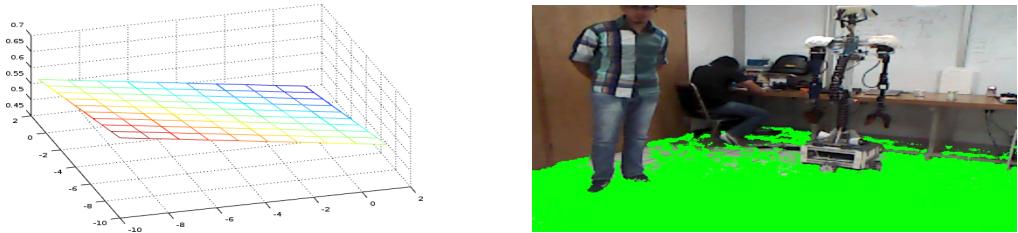
Por lo tanto con tres puntos seleccionados aleatoriamente tenemos un modelo de plano generado aleatoriamente. Lo cual es el primer requerimiento para el algoritmo.

Posteriormente se comparó el resto de la información con ese modelo propuesto aleatoriamente y verificamos que tan buen modelo es. Para ello calculamos la distancia euclídea de cada punto al plano propuesto, si la distancia es menor que cierto umbral lo consideramos como parte del modelo e incrementamos el numero de puntos que empatan con el modelo inliers:

Del mismo modo se buscaba que el numero de inliers que empataran con el modelo fuera al menos el 60 % del total de datos. En los resultados se trata este tema con mayor profundidad.

Observamos que el algoritmo opera bien con un numero máximo de iteraciones = 70, y con un umbral de error de 2 cm.

Por ahora el algoritmo no restringe su búsqueda a planos horizontales o verticales, quedá como trabajo futuro el desarrollo de una búsqueda selectiva de planos con características similares a una mesa, destacando características de horizontalidad y altura.



(a) Grafico de un plano en tercera dimensión-(b) Segmentación de un plano utilizando el sensor kinect.

Figura 3.2: Planos y su representación en 3D.

3.2. Extracción de objetos

Una vez que se encontró la nube de puntos perteneciente al modelo, se prosiguió a ubicar espacialmente los objetos que se encontraban sobre el plano. Se realizó la extracción de objetos con las siguientes restricciones:

- $p_n = \text{null} \forall p_n \in \Pi$
- $p_n = \text{null} \forall p_n.z < \text{Height}(\Pi)$
- $p_n = \text{null} \forall p_n.x > 0,50[m]$

De otro modo esto sería igual a:

- Extracción de los puntos pertenecientes al plano.
- Extracción de los puntos por debajo del plano.
- Extracción de los puntos más allá de 40 respecto al punto más cercano en x.

Posterior a este proceso tenemos como resultado un conjunto de puntos reflejados sobre un objeto sólido con las distancias proyectadas sobre los tres ejes coordinados respecto del robot Justina.



Figura 3.3: Extracción de información del objeto.

3.3. Aproximación de orientación de objetos con PCA

El análisis de componentes principales se usa abundantemente en muchas formas de análisis de datos. En este caso se utilizará como una metodología para reducir un conjunto de datos a una dimensión inferior. Más concretamente la forma en que podemos obtener una base ortogonal del conjunto de datos, que represente la distribución de los puntos proyectados sobre el objeto. Esta base, representará la dirección de la distribución de los puntos del objeto, y por tanto encontraremos la dirección del objeto mismo respecto al robot.

3.3.1. Adquisición de datos y cálculo de la media

La adquisición de datos se realizó mediante la extracción de los puntos pertenecientes al objeto. Cada uno de los datos perteneciente al conjunto de puntos posee tres características:

- La distancia al objeto proyectada sobre el eje x
- La distancia al objeto proyectada sobre el eje y
- La distancia al objeto proyectada sobre el eje z

Por lo tanto, de una nube de puntos como la mostrada en la figura () podemos extraer medidas de dispersión, y medidas de tendencia central. Se partió con el cálculo de la media de los datos. $\bar{X}, \bar{Y}, \bar{Z}$. El punto $P(\bar{X}, \bar{Y}, \bar{Z})$

representa el centroide del objeto. Conociendo este punto podemos conocer la ubicación espacial del objeto respecto del robot y determinar si es manipulable (si se encuentra dentro del área de trabajo) y conocer las coordenadas a donde deberíamos llevar el efecto final de nuestro manipulador para manipularlo.

Para realizar el cálculo de la media se cuantificó el número de puntos pertenecientes al objeto, (n). Posteriormente, para todos los puntos pertenecientes al conjunto n se sumaron sus valores en $\bar{X}, \bar{Y}, \bar{Z}$, respectivamente. Y posteriormente se dividieron entre n .

```

for cada punto in datos do
     $X_{total} = X_{total} + dato_x$ 
     $Y_{total} = Y_{total} + dato_y$ 
     $Z_{total} = Z_{total} + dato_z$ 
end for

 $\bar{X} = \frac{X_{total}}{n}$ 
 $\bar{Y} = \frac{Y_{total}}{n}$ 
 $\bar{Z} = \frac{Z_{total}}{n}$ 

return  $P(\bar{X}, \bar{Y}, \bar{Z})$ 
```



Figura 3.4: Segmentación de un objeto y su correspondiente centroide (punto verde).

3.3.2. Cálculo de la matriz de covarianza

Dada una variable estadística n-dimensional ($X_1, X_2, X_3, \dots, X_n$), llamaremos matriz de covarianzas a la matriz cuadrada, $n \times n$, que disponga en su diagonal principal de las varianzas de cada una de las distribuciones marginales unidimensionales, y en los elementos no-diagonales (i, j) de las correspondientes covarianzas entre cada dos variables S_{ij} .

Para ello se hizo uso de las expresiones matemáticas correspondientes:

$$var(X) = \frac{\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})}{n} \quad (3.1)$$

$$cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{n} \quad (3.2)$$

A continuación se anexa en código correspondiente al cálculo de las varianzas y covarianzas.

```

1 //Calculate covariance matrix
2 for(int j = 0; j < object.rows; j++)
3     for (int i = 0; i < object.cols; i++)
4     {
5         px = object.at<cv::Point3f>(j, i);
6         if ( px != cv::Point3f(0.0, 0.0, 0.0) && px != cv::Point3f(0, 255, 0))
7         {
8             var_x += pow( (px.x - centroid[0]), 2 );
9             var_y += pow( (px.y - centroid[1]), 2 );
10            var_z += pow( (px.z - centroid[2]), 2 );
11            cov_xy += ( px.x - centroid[0] )*( px.y - centroid[1] );
12            cov_xz += ( px.x - centroid[0] )*( px.z - centroid[2] );
13            cov_yz += ( px.y - centroid[1] )*( px.z - centroid[2] );
14            n++;
15        }
16    }
17
18 var_x /= n;
19 var_y /= n;
20 var_z /= n;
21
22 cov_xy /= n;
23 cov_xz /= n;
24 cov_yz /= n;
25
26 cov_matrix << var_x, cov_xy, cov_xz,
27             cov_xy, var_y, cov_yz,
28             cov_xz, cov_yz, var_z;
29

```

Listing 3.1: Cálculo de varianzas y covarianzas

Posteriormente se propuso a encontrar los valores de los vectores propios asociados a la matriz de covarianzas. En la literatura existen múltiples métodos numéricos para la solución de esta problemática, como la puesta a prueba de dichos algoritmos no es el propósito de este trabajo se optó por utilizar la librería **eigen**. Por tanto, una vez construida la matriz de covarianzas M, se utiliza la instrucción:

```
1 Eigen::SelfAdjointEigenSolver<Eigen::MatrixXf> eig(cov_matrix);
2
```

Listing 3.2: Cálculo de varianzas y covarianzas

La instrucción *Eigen::SelfAdjointEigenSolver;Eigen::MatrixXf;* nos devuelve un objeto *eig* cuyos atributos son *eigenvalues()* y *eigenvectors()*. Con ayuda de esta librería podemos obtener fácilmente los eigenvectores de la matriz M. Es importante mencionar que el atributo *eigenvectors()* del objeto *eig* se encuentra normalizado; por lo tanto es conveniente darle un escalamiento relacionado con la dispersión de los datos. De este modo se optó por multiplicar cada uno de los eigenvectores por su correspondiente valor de desviación estándar.



Figura 3.5: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

Capítulo 4

Manipulador

En esta sección se hablará de las caracteristicas mecánicas del brazo robótico ensamblado en el robot de servicio Justina. Se describirán los elementos que componen el brazo robótico haciendo enfasis en la ubicación de los actuadores y como influye esto en el modelo matematico del mismo.

4.1. Documentación servos dynamixel

Los servomotores Dynamixel son un sistema de actuador inteligente desarrollados con el propósito de funcionar como uniones de conexión en un robot o estructura mecánica. Los servomotores Dynamixel están diseñados para ser modulares y soportar la conexión en cadena en cualquier robot o diseño mecánico. Este tipo de servomotores es popular por los beneficios que ofrece: movimientos robóticos potentes y flexibles.

El conjunto de servomotores Dynamixel son un grupo de actuadores de alto rendimiento con reductor, controlador y un protocolo de red completamente integrados en un módulo de servomotor programable y en red. El estado del actuador se puede leer y monitorear a través de un flujo de paquetes de datos. [22]

Los servomotores de la marca Dynamixel, cuentan con diferentes gamas de productos de acuerdo a las necesidades de cada proyecto. El brazo robótico referente a este documento está constituido únicamente por motores de la gama MX, de la cual se describen las características más sobresalientes a continuación.

4.1.1. Motores dynamixel Serie MX

Los actuadores dynamixel de la serie MX pertenecen a la última generación de actuadores de la compañía Robotis Dynamixel. Están equipados con un procesador Cortex M3 de 32 bits a 72 mhz, un encoder magnético sin contacto con una resolución 4 veces mayor que la serie AX / RX, además ofrecen una velocidad de comunicación de hasta 3 mbps con el nuevo bus TTL 2.0. Cada servomotor cuenta con la capacidad de monitorear su velocidad, temperatura, posición del eje, voltaje y carga.

Por otro lado los motores Dynamixel de la serie MX cuentan con la implementación de un algoritmo de control PID para mantener la posición del eje. Las ganancias del algoritmo de control PID se pueden ajustar individualmente para cada servo, lo que le permite controlar la velocidad y la fuerza de la respuesta del motor. Todos los servos de la serie MX usan un voltaje nominal de 12v. La característica más importante de esta serie de servomotores es que la gestión de la información proveniente del sensor y la ejecución del algoritmo de control de posición son manejados por el microcontrolador integrado del servo. Este enfoque distribuido deja a su controlador principal libre para realizar otras funciones.

En lo que respecta a la parte mecánica, todos los servos DYNAMIXEL son compatibles con una amplia variedad de bridales, acoplamientos y sujetadores lo cual facilita la construcción de cualquier configuración deseada. Este conjunto de elementos mecánicos permite una conexión directa a cualquier modelo de servomotores DYNAMIXEL, lo que le permite una gran variedad de configuraciones para cualquier necesidad. Dada la gran variedad de configuraciones, este tipo de elementos de unión y fijación resultó de gran ayuda mecánica al momento de construir el brazo robótico donde se implementaron los algoritmos descritos en el presente trabajo.

Resumiendo las características de los servomotores de la serie MX, podemos enlistar las siguientes:

- Movimiento sin contacto y detección de posición.
- Interfaz de comunicación TTL estándar.
- Comunicación de alta velocidad de hasta 4.5 Mbps.
- Control PID para autocorrección en posicionamiento.

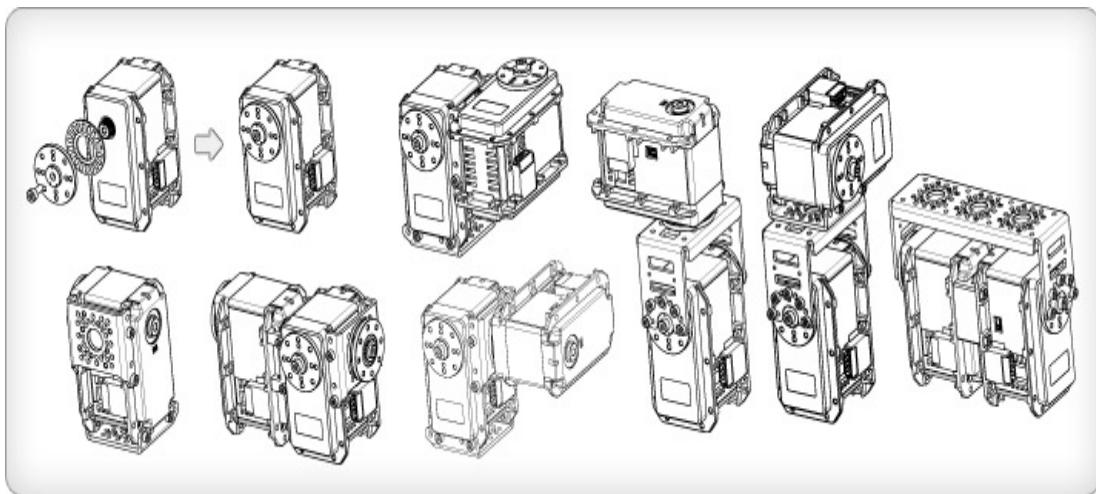


Figura 4.1: Variedad de ensambles para servomotores.

- Control de par basado en corriente (4096 steps, 2.69mA/step)
- Ahorro de energía (corriente reducida de 100 mA a 40 mA)

Si desea controlar este servo DYNAMIXEL desde un equipo de computo es necesario contar con un dispositivo que facilite la comunicación entre este y el microprocesador de cada uno de los servomotores, el dispositivo que cumple con esta funcionalidad es el USB2Dynamixel.

El controlador USB2Dynamixel se conecta al puerto USB en una computadora y tiene un puerto de 3 y 4 pines para conectar Dynamixels. El USB2Dynamixel se puede usar con controladores como CM-5 y CM-2 que usan comunicación en serie, además soporta la comunicación en cadena propia de los servomotores Dynamixel. Es importante mencionar que este dispositivo posee un interruptor para seleccionar entre diferentes protocolos de comunicación: RS-232, RS-485, TTL. Por tanto consta de:

- Conector 3P: puerto de comunicación para nivel TTL (para control de serie AX y MX)
- Conector 4P: puerto de comunicación RS-485 (para control de serie DX, RX, EX y MX)

Los motores que componen esta gama comparten características, las cuales se mencionan a continuación:

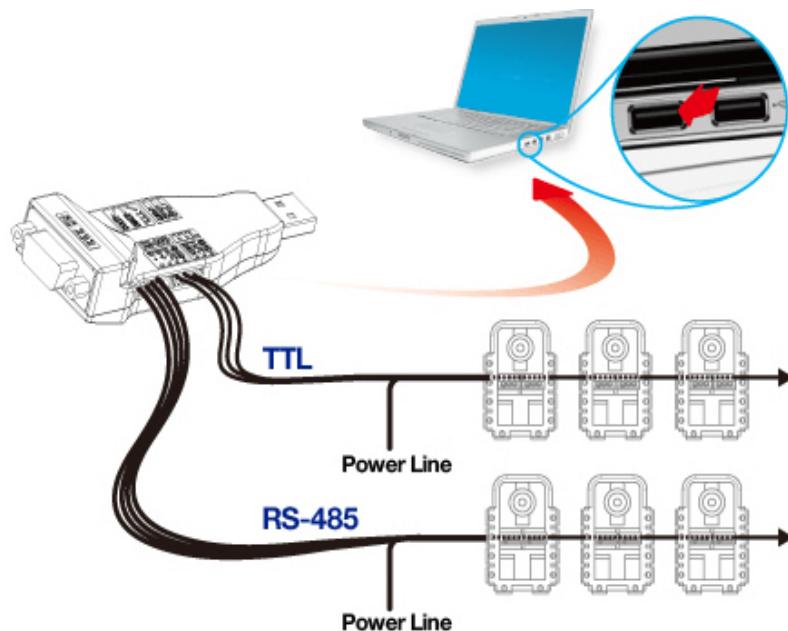


Figura 4.2: Dispositivo de comunicación USB a protocolo serial.

MOTORES GAMA MX			
Voltaje de operación	14.8v	12v	11.1v
Protocolo	TTL Serial Asincrono		
Velocidad del puerto	8000bps - 3Mbps		
Retroalimentación de posición	SI		
Retroalimentación de temperatura	SI		
Control PID	SI		
Material	Carcasa platica y reducción metálica		
Motor	Maxon RE-MAX		

Tabla 4.1: Características servomotores MX.

Los modelos de motores que componen esta gama difieren en características tales como las dimensiones, el par máximo a rotor bloqueado, la corriente máxima, o la velocidad máxima. A continuación se en listan las características más sobresalientes para cada modelo de motor.

4.1.2. Características MX-106

MX-106			
Voltaje de operación	14.8v	12v	11.1v
Par de bloqueo	102kg.cm	85.6kg.cm	81.5kg.cm
Velocidad sin carga	55rpm	45rpm	41rpm
Peso	153g		
Tamaño	40.2 x 65.1 x 46 mm		
Resolución	0.088 grado/valor de registro		
Relación de reducción	1/200		
Corriente máxima	5.2A @ 12V		

Tabla 4.2: Características dynamixel modelo MX-106.



Figura 4.3: Servomotor dynamixel modelo MX-106.

4.1.3. Características MX-64

MX-64			
Voltaje de operación	14.8v	12v	11.1v
Par de bloqueo	74kg.cm	61kg.cm	56kg.cm
Velocidad sin carga	78rpm	63rpm	58rpm
Peso	126g		
Tamaño	40.2 x 61.1 x 41 mm		
Resolución	0.088 grado/valor de registro		
Relación de reducción	1/200		
Corriente máxima	4.1A @ 12V		

Tabla 4.3: Características Dynamixel modelo MX-64



Figura 4.4: Servomotor dynamixel modelo MX-64.

4.1.4. Características MX-28

MX-28			
Voltaje de operación	14.8v	12v	11.1v
Par de bloqueo	31.6kg.cm	25.5kg.cm	23.4kg.cm
Velocidad sin carga	67rpm	55rpm	50rpm
Peso	72g		
Tamaño	35.6 x 50.6 x 35.5 mm		
Resolución	0.088 grado/valor de registro		
Relación de reducción	1/193		
Corriente máxima	1.4A @ 12V		

Tabla 4.4: Características Dynamixel modelo MX-64



Figura 4.5: Servomotor dynamixel modelo MX-64.

Por otro lado es importante conocer la estructura de transmisión de la información de un servomotor con el dispositivo controlador. Para ello los servomotores dynamixel cuentan con un protocolo de comunicación serial asincrono, el cual transtite una serie de datos en una trama, como se ilustra en la figura[].



El significado de cada byte que compone al paquete datos es el siguiente:

- **0xFF 0xFF** : Es la instrucción que marca el inicio del paquete.
- **ID** : Es el ID de cada servomotor particular en la cadena.
- **LENGTH** : Es el tamaño del paquete. El tamaño es calculado como el numero de parametros más dos (N+2).
- **INSTRUCTION** : Indica el tipo de instrucción que ejecutará el servomotor dynamixel.
 - PING
 - READ_DATA
 - WRITE_DATA
 - REG_WRITE
 - ACTION
 - RESET
 - SYN_WRITE
- **PARANETER 0...N** : Este parámetro se usa cuando la instrucción requiere datos auxiliares.
- **CHECK SUM** : Se usa para verificar si el paquete de datos se dañó durante la comunicación.

A continuación se anexa una tabla con las caracteristicas más relevantes de los registros que posee cada uno de los revomotores dynamixel.

AREA	Dirección	Nombre	Descripción	Acceso
	3	ID	Identificador del motor	L/E
	4	Baud Rate	Velocidad transmisión datos dynamixel	L/E
	10	Drive mode	Configuraciones en modo dual	L/E
	14	Max Torque(L)	Byte bajo del registro Máx torque	L/E
	15	Max Torque(H)	Byte alto del registro Máx torque	L/E
	24	Torque Enable	On Off Torque	L/E
	26	D gain	Ganancia de control derivativa	L/E
	27	I gain	Ganancia de control integral	L/E
	28	P gain	Ganancia de control proporcional	L/E
	30	Goal position(L)	Byte bajo del registro para una posición deseada	L/E
	31	Goal position(H)	Byte alto del registro para una posición deseada	L/E
	32	Moving Speed(L)	Byte bajo del registro para una velocidad deseada	L/E
	33	Moving Speed(H)	Byte alto del registro para una velocidad deseada	L/E
	34	Torque Limit(L)	Byte bajo del registro para un par deseado	L/E
	35	Torque Limit(H)	Byte alto del registro para un par deseado	L/E
	36	Present position(L)	Byte bajo del registro de la posición presente	L
	37	Present position(H)	Byte alto del registro de la posición presente	L
	40	Present load(L) 45	Byte bajo del registro de la carga presente	L
	41	Present load(H)	Byte alto del registro de la carga presente	L

Tabla 4.5: Tabla de registros para servomotores Dynamixel de la gama MX.

4.1.5. Características de conexión

Se ha mencionado con anterioridad que el brazo robótico con el cual se realizaron las pruebas correspondientes para este trabajo consta de 10 servomotores dynamixel de la gama MX y de diferentes modelos. Para facilitar la comunicación entre los diferentes modelos de servomotores se utilizó un método de conexión *daisy chain* en el cual permite una conexión en cadena entre servomotores, asignandole un ID a cada servomotor y realizando la comunicación a través de un solo puerto.

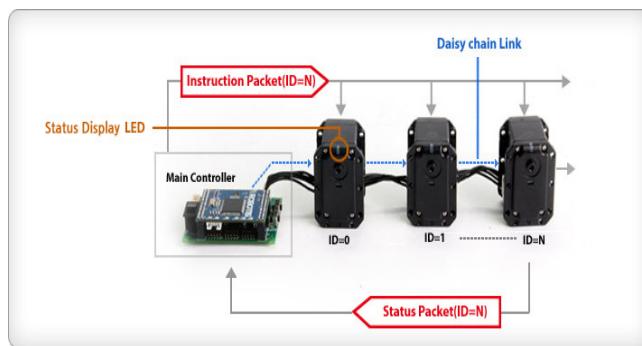


Figura 4.6: Servomotores dynamixel en conexión daisy chain.

El sistema *daisy chain* es un esquema de conexiones que forman una sucesión de enlaces, tal que el dispositivo A se encuentra conectado al dispositivo B y este, a su vez, se encuentra conectado a un dispositivo C y así sucesivamente. Es importante señalar que en este tipo de conexión los dispositivos no forman redes, en tal caso el dispositivo C no podría estar conectado al dispositivo A. Otro aspecto importante de resaltar es que en este tipo de conexiones al no poder formar redes la comunicación se realiza dispositivo a dispositivo, por tanto los dispositivos ultimos en la cadena pueden presentar retraso o fallas eléctricas con respecto de los dispositivos primeros en orden de la cadena.

4.2. Descripción de los elementos del sistema de manipulación

El sistema de Hardware del brazo robótico está compuesto por un total de 10 servomotores Dynamixel fabricados por la compañía Robotis [22]. Eta

compañía cuenta con diversas gamas de modelos de servomotores según las necesidades de la aplicación. A continuación se describirá el sistema en orden descendente.

En la parte superior del brazo robótico se encuentran dos servomotores Dynamixel MX-106 conectados como maestro-esclavo, configurados en este modo trabajan de manera conjunta aumentando el par unitario de cada uno de los servomotores. Dadas las características antes descritas en la tabla 4.2 podemos observar que la configuración de servomotores entrega un par de torsión máximo a rotor bloqueado de $16,8\text{N.m}$ @ 12V. El sentido de giro positivo del arreglo de servomotores es el mostrado en la figura[4.8].

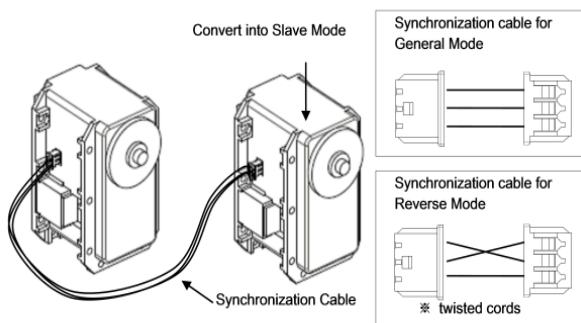


Figura 4.7: Configuración maestro-esclavo en servomotores dynamixel.

tamente controlado por la señal PWM del maestro transmitida a través del cable de sincronización. Es importante mencionar que la información de la posición, velocidad y corriente deseada es ignorada; puesto que esta información depende únicamente del maestro.

En la configuración maestro-esclavo el esclavo puede configurarse para adoptar un sentido de giro inverso en caso que el acoplamiento mecánico así lo requiera. Dentro de este modo el servomotor esclavo tendrá la misma velocidad que el motor maestro, solo diferenciado por el sentido inverso de giro.

Posteriormente se encuentra un servomotor modelo MX-106, ubicado en una disposición horizontal. Unido a este servomotor se encuentra un eslabón obtenido mediante técnicas de manufactura aditiva. Dado que la construcción del brazo robótico esta pensada desde el punto de vista antropomórfico este conjunto de servomotores, el maestro-esclavo y el servomotor MX-106

La configuración maestro-esclavo es un método de control simultáneo para dos servomotores Dynamixel, esta configuración es sumamente útil cuando se trata de construir una junta cuyo eje de acción es coincidente. Los motores respectivos maestro y esclavo deben estar conectados mediante un cable de sincronización, como se muestra en la figura[4.7]. El servomotor esclavo es direc-

describen los grados de libertad que emulan el movimiento de un hombro humano.

El eslabón numero 1 obtenido mediante técnicas de manufactura aditiva funge como elemento de unión entre servomotores. Este elemento de unión cuenta con un peso total de **** Al final de este elemnto de unión se encuentra un servomotor MX-64 cuyo eje de acción es perpendicular respcto a la base el brazo. Directamente acoplado a este servomotor se encuentra otro servomotor modelo MX-106. Estos dos servomotores realizan el símil con un codo humano.

Continuando en orden descendente, se encuentra un segundo eslabón utilizado como elemento de unión. En el extremo de este elemento se encuentra un servomotor MX-106, posteriormente un MX-64 y por último un servomoto modelo MX-28. Estos tres elementos construyen un sistema similar al de la muñeca de un brazo humano.

Por último, se encuentra el efecto final compuesto de dos actuadores MX-28 ubicados en disposición horizontal. En el eje de acción de cada uno de estos se encuentra una pieza obntenida de impresión 3D que funge como sujetador para manipular objetos.

Como podemos observar el brazo robótico consta de un total de 10 servomotores conectados en *daisy chain*. Dentro de este trabajo se realizó en analisis de la transmisión de datos para cada uno de los servomotores. Para este analisis se partió de una configuración de conexión previamente establecida, como se muestra en la figura []. Posteriormente se realizaron un total 1000 lecturas del registro *Present Position* para cada uno de los actuadores. Se observó que la cantidad de datos perdidos era proporcional al número que el actuador tenía asignado en la cadena. De esta manera, se realizó la hipotesis



Figura 4.8: Diseño asistido por computadora del brazo robótico del robot Justina.

que los datos se perdían a lo largo de la cadena de actuadores. Posteriormente se realizaron mediciones de voltaje a lo largo de la cadena, se observó una caída de 0.5V entre el actuador denominado A y los actuadores I-J.

Los resultados obtenidos nos indicaron dos posibles ejes de acción. El primero de ellos, por facilidad de implementación, fue cambiar la velocidad de transmisión de datos para cada uno de los servomotores, ya que una velocidad alta podría implicar una colisión de datos en la transmisión, lo cual conllevaría a el fenómeno que observamos. El segundo eje de acción fue el análisis del consumo de corrientes en cada uno de los actuadores. Este evento pudo haber ocasionado la caída de voltaje y posteriormente la perdida de datos al no estar adecuadamente alimentados los dispositivos.

**** Cadena completa y muestra de las conexiones eléctricas (comparación antes y después.)

4.3. Cinemática directa

El problema de cinemática directa consiste en conocer la posición (x, y, z) del efecto final dada una determinada configuración de ángulos para el conjunto de actuadores que forman el sistema mecánico. Para ello se describe la posición del efecto final en términos de las transformaciones existentes entre este y la base fija del manipulador.

4.3.1. Teoría trasformaciones y medidas del brazo robótico

Se partió del sistema mecánico previamente construido del brazo robótico del robot de servicio Justina. Para ello se realizaron mediciones de distancias entre los respectivos ejes de acción de cada uno de los actuadores a fin describir los desplazamientos de los sistemas de referencia en el brazo robótico.

***Foto brazo real con medidas

Es importante mencionar que para la descripción de las transformaciones en sistema ROS, podemos construir la matriz de rotación de tal manera que incluyamos las rotaciones y desplazamientos en los tres ejes coordinados. Esta característica nos proporciona una ventaja sobre las técnicas de descripción de cinemática directa convencionales, tal es el caso del Método Denavit-Hartenberg donde el sistema de transformaciones debe ser descrito utilizando, únicamente, dos rotaciones y dos translaciones según sus convenciones.

Es importante mencionar esta característica puesto que en algunos casos la concepción Denavit-Hartenberg encuentra limitaciones para describir algunas transformaciones. ***cite spog[] issues

$$T_n^{n-1} = T_{Z_{n-1}}(d_n) * R_z(\theta_n) * T_{x_n}(a_n) * R_{x_n}(\alpha) \quad (4.1)$$

$$T_n^{n-1} = \begin{bmatrix} \cos \theta_n & -\sin \theta_n \cos \alpha_n & \sin \theta_n \sin \alpha_n & a_n \cos \theta_n \\ \sin \theta_n & \cos \theta_n \cos \alpha_n & -\cos \theta_n \sin \alpha_n & a_n \sin \theta_n \\ 0 & \sin \alpha_n & \cos \alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

Como podemos observar, en la convención de transformaciones Denavit-Hartenberg únicamente requerimos determinar el valor de cuatro variables, dos rotaciones (α, θ) y dos translaciones (a, d).

El sistema de descripción de transformaciones desarrollado por ROS, presenta la ventaja que nos permite describir la trasformación de una manera más detallada. Sin embargo la solución implementada no siempre suele ser la más óptima, encuanto al manejo de la información por ser una mayor cantidad de datos.

Para ello se parte de la matriz de rotación compuesta roll, pitch, yaw.

$$R_1^0 = R_{z,\phi} * R_{y,\theta} * R_{x,\psi} \quad (4.3)$$

$$R_1^0 = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi \\ 0 & \sin \psi & \cos \psi \end{bmatrix} \quad (4.4)$$

$$= \begin{bmatrix} \cos \phi \cos \theta & -\sin \phi \cos \psi - \cos \phi \sin \theta \sin \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \sin \phi \cos \theta & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi \\ -\sin \theta & \cos \theta \sin \psi & \cos \theta \cos \psi \end{bmatrix} \quad (4.5)$$

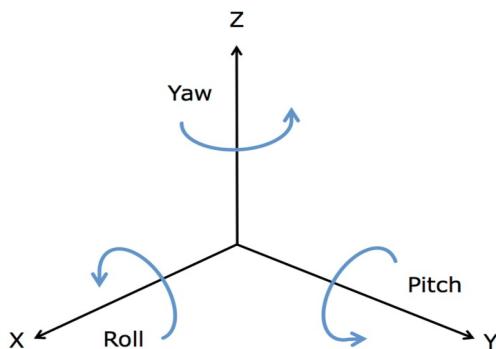


Figura 4.9: Rotaciones roll, pitch, yaw.

4.3.2. Implementación (Descripción con ROS)

Se utilizó un visualizador en 3D para observar el modelo del brazo y verificar el correcto movimiento mecánico del brazo. En tal caso se hizo uso de los paquetes proporcionados por *ROS* para la elaboración de modelos de robots. La descripción del modelo del robot se realizó en una estructura en formato XML. Esta descripción permite describir las transformaciones existentes entre cada uno de los frames significativos en el robot, así como permite cargar algún modelo *CAD* (*Computer Assisted Design*) de las piezas que componen al robot.

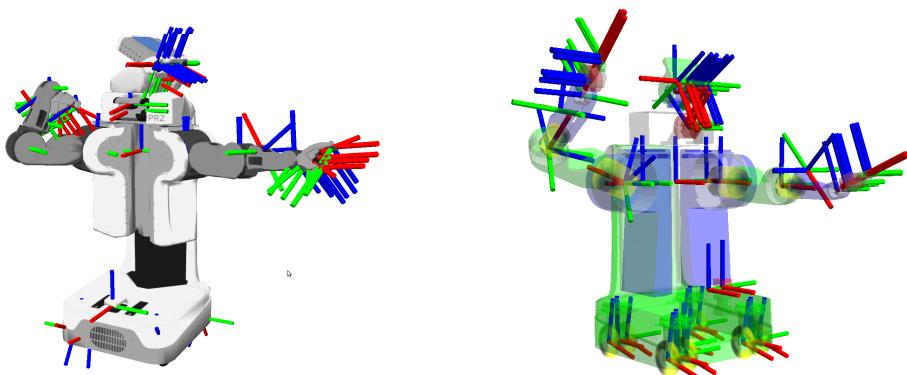
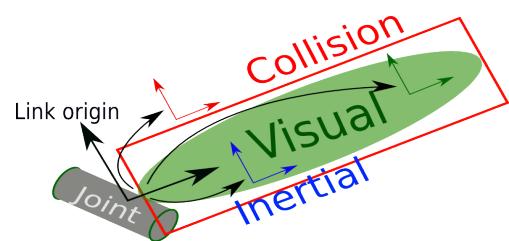


Figura 4.10: Ejemplo de un modelo de robot y sus respectivas transformaciones.

ROS proporciona una biblioteca de software únicamente dedicada al manejo de información de las transformaciones de un robot, la biblioteca de software lleva el nombre de *tf*. En un nivel práctico, un árbol de transformación define los desplazamientos en términos de translación y rotación entre diferentes marcos de coordenadas. La biblioteca *tf* usa una estructura en árbol para garantizar que solo haya un recorrido único que vincule dos sistemas coordinados entre sí y asume que todas las transformaciones del árbol se dirigen desde los nodos primarios a secundarios.

Para describir una transformación se requiere definir primero un conector, que funcione como cada uno de los elementos físicos, posteriormente la transformación se encargará de relacionar cada uno de estos conectores. Conceptualmente, cada nodo en el árbol de transformación



corresponde a un sistema de coordenadas el cual representa cada uno de los conectores; por otra parte cada enlace de conexión corresponde a la transformación que debe aplicarse para pasar del nodo actual a su hijo.

Para crear el árbol de transformaciones del manipulador de 7DOF, comenzamos por crear un sistema de referencia en la base del brazo, el cual servirá como sistema de referencia fijo, lo llamamos *base_ra_arm*. Posteriormente se fueron agregando eslabones según la configuración real del brazo. Para llegar a la configuración se partió de la siguiente restricción: el eje Z debe coincidir con el eje de giro del actuador, en posición y en sentido positivo, como se muestra en la figura.

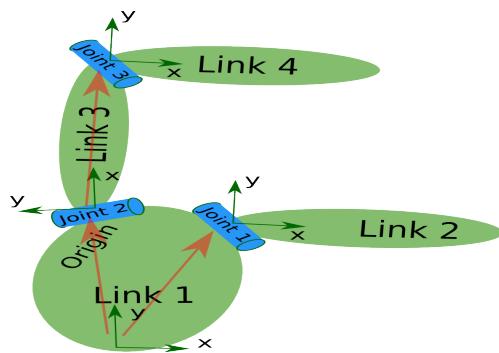


Figura 4.12: Ejemplo de sistema de transformaciones (sistemas de referencia y links).

Para crear esta configuración entre los diferentes sistemas de referencia, primero debemos decidir qué nodo será el padre y cuál será el hijo. Esta distinción es importante porque asume que todas las transformaciones se mueven de padres a hijos. Elegimos el sistema de referencia *base_ra_arm* como principal, ya que a medida que se agregan otras piezas: motores, eslabones y conectores, tendrá más sentido que se relacionen con un sistema fijo. Esto significa que la transformada asociada con el efecto final y el sistema *base_ra_arm* puede ser obtenida mediante el árbol de transformaciones configurado. La conversión de la posición del sistema de referencia ubicado en el efecto final al sistema de referencia fijo ubicado en la base del brazo puede ser obtenida mediante una llamada a la biblioteca tf.

Con la información previamente obtenida de las dimensiones del brazo robótico se construyó la siguiente tabla donde se muestra la información de los desplazamientos y rotaciones de los respectivos sistemas de referencia.

	TRANSLACIÓN			ROTACIÓN		
	x	y	z	roll	pitch	yaw
1	0.065	0.0	0.000	$\pi/2$	0.000	0.0
2	0.215	0.0	0.000	0.000	$\pi/2$	0.0
3	0.000	0.0	0.060	$-\pi/2$	$-\pi/2$	0.0
4	0.190	0.0	0.000	$\pi/2$	0.000	$\pi/2$
5	0.000	0.0	0.036	$-\pi/2$	$-\pi/2$	0.0
6	0.095	0.0	0.000	$\pi/2$	0.000	$\pi/2$

Tabla 4.6: Tabla de parametros de transformaciones entre sistemas de refencia para el brazo robotico de 7DOF.

Con la información obtenida de las dimensiones del brazo robótico real pudimos realizar un modelo virtual del brazo utilizando un archivo en formato XML donde describimos las trasfomaciones entre los diferentes sistemas de referencia y los elemets de conexión obtenidos de modelos CAD. El resultado se puede observar en la figura[4.13]

El formato de descrpción virtual de un modelo de robot en formato XML se compone de *links* y *joints*. Un link describe las características de un elemento de unión, estas pueden ser mecánicas, de material, de posición, etc. Un *joint* por otra parte describe la realción existente entre los orígenes de dos links, dicho de otro modo describe la transformación existente entre los sistemas de refencias (*orígenes*) de dos links.

```

1 <link name="base_ra_arm">
2   <visual>
3     <origin xyz="-0.01 0.0 0.0" rpy="1.5707 0.0 1.5707"/>
4     <geometry>
5       <mesh filename="package://knowledge/hardware/stl/brazo/mx106_2.stl"/>
6     </geometry>
7     <material name="black_gray"><color rgba="0.2 0.2 0.2 1"/></material>
8   </visual>
9 </link>
10
11 <link name="ra_link0">
12   <visual>
13     <origin xyz="0.0 0.0 0.005" rpy="0.0 -1.5707 0.0"/>
14     <geometry>
15       <mesh filename="package://knowledge/hardware/stl/brazo/mx106_s.stl"/>
16     </geometry>
17     <material name="gray_black"><color rgba="0.3 0.3 0.3 1"/></material>
18   </visual>
19 </link>
20
21 <link name="ra_link1">
22   <visual>
23     <origin xyz="0.0 0.0 0.0" rpy="1.5707 -1.5707 0.0"/>
24     <geometry>
25       <mesh filename="package://knowledge/hardware/stl/brazo/bone_1.stl"/>
26     </geometry>
27     <material name="ra_material"><color rgba="0.9 0.85 0.75 1"/></material>
28   </visual>
29 </link>
30 .
31 .
32 .
33 .
34 .
35
36 <joint name="ra_1_joint" type="revolute">
37   <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
38   <parent link="base_ra_arm"/>
39   <child link="ra_link0"/>
40   <limit effort="0.0" lower="0.0" upper="0" velocity="0.0"/>
41   <axis xyz="0 0 1"/>
42 </joint>
43
44 <joint name="ra_2_joint" type="revolute">
45   <origin xyz="0.064 0.0 0.0" rpy="1.5707 0.0 0.0"/>
46   <parent link="ra_link0"/>
47   <child link="ra_link1"/>
48   <limit effort="0.0" lower="0.0" upper="0" velocity="0.0"/>
49   <axis xyz="0 0 1"/>
50 </joint>
51

```

Listing 4.1: Descripción de sistema de manipulación

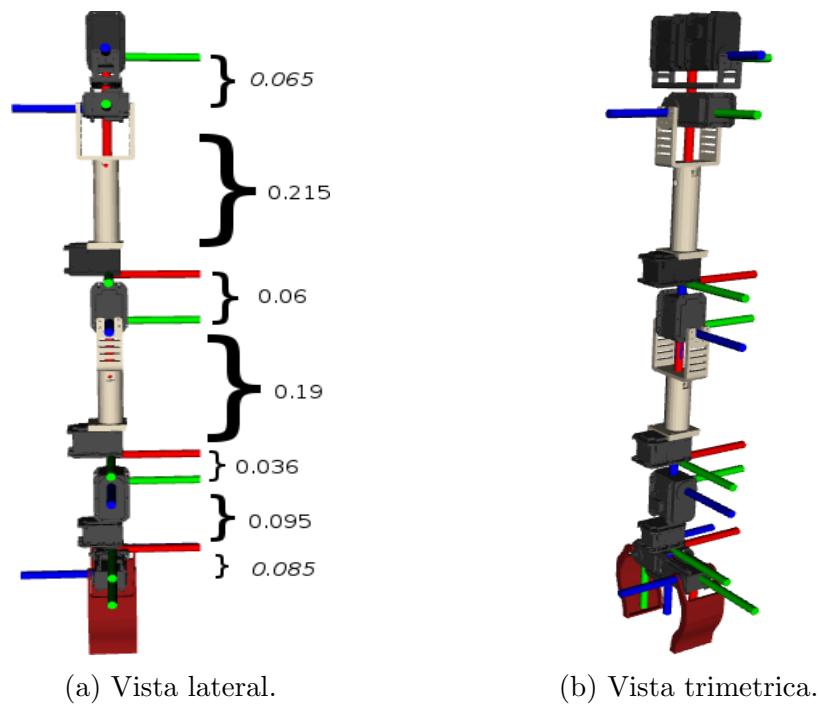


Figura 4.13: Modelo virtual del brazo robótico.

4.4. Cinemática inversa

Capítulo 5

Integración

- 5.1. Máquinas de estados**
- 5.2. ROS**
- 5.3. Constitución del robot de servicio Justina**

Capítulo 6

Resultados y conclusiones

En esta sección analizaremos los resultados obtenidos al probar el algoritmo RANSAC para encontrar planos modificando el numero de iteraciones. Observaremos el comportamiento del error y el tiempo de ejecución a fin de encontrar el número óptimo de iteraciones para este algoritmo.

El proceso para el análisis de datos fue el siguiente: se propuso un modelo de plano conocido por el usuario. Dicho modelo se obtuvo a partir del conocimiento de la altura del plano en este caso una mesa. Posteriormente se cuantificó la cantidad de puntos que entraban en este modelo ideal y se tomó como base para la medición de errores. Continuando con el procedimiento se modificó el algoritmo para realizar un número determinado de iteraciones (600, 200, 100, 50, 30, 24 y 20) y se midió el error relativo y el tiempo de ejecución, en cada uno de estos procedimientos.

Para probar el algoritmo con 600 iteraciones se tomaron 50 muestras los resultados se pueden observar en la siguiente gráfica.



Figura 6.1: Segmentación de un objeto y sus correspondientes ejes principales (rojo).



Figura 6.2: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

Se tomaron 50 muestras para el algoritmo modificado con 200 iteraciones. Los resultados se pueden observar en la siguiente gráfica.

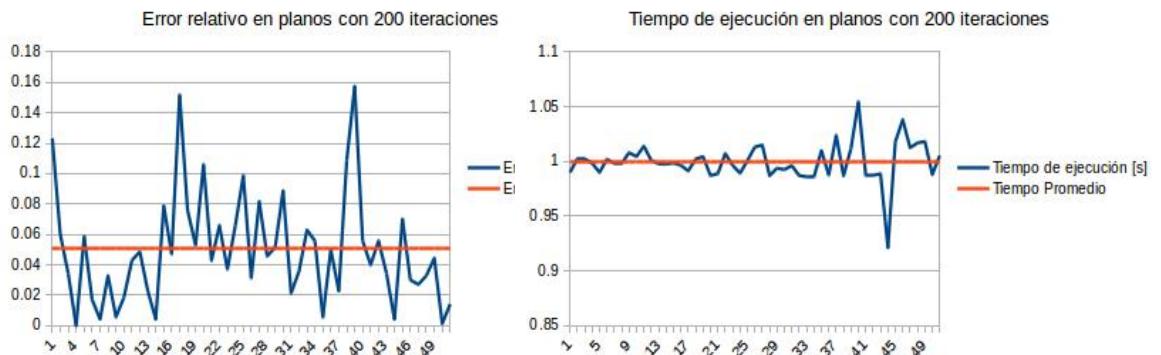


Figura 6.3: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

Se continuó con el mismo principio para el algoritmo modificado con 100, 50, 30, 24 y 20 iteraciones respectivamente. Los resultados se pueden observar en las gráficas anexadas a continuación.

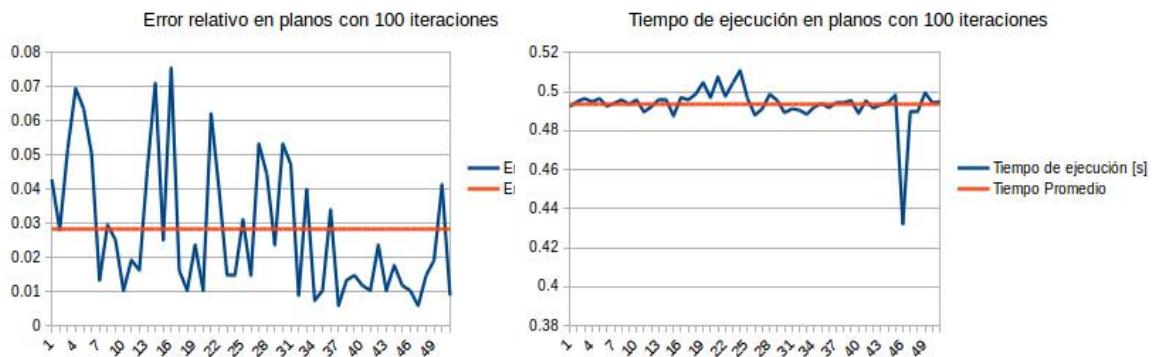


Figura 6.4: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

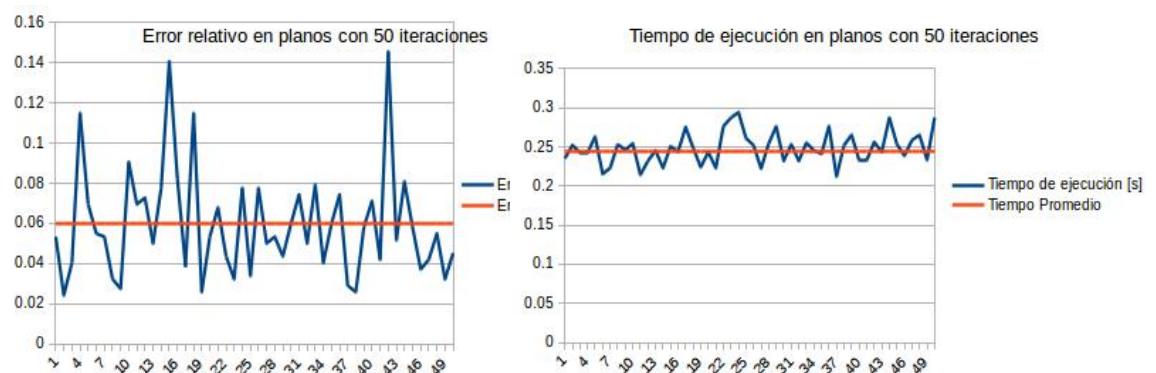


Figura 6.5: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

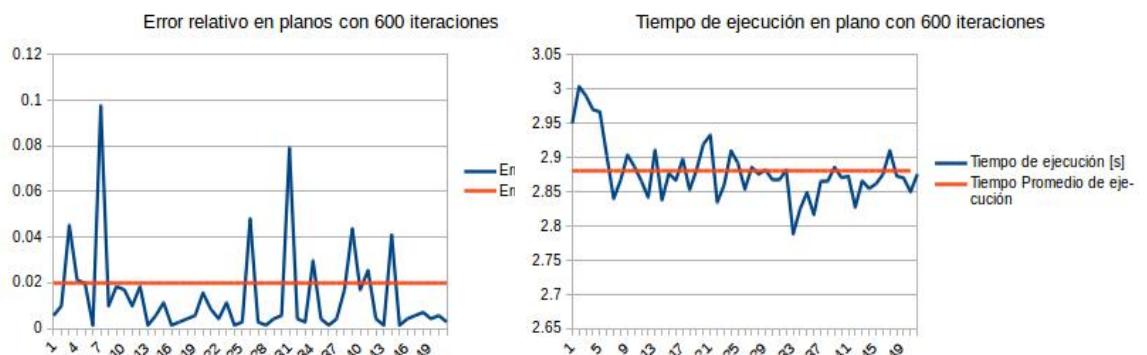


Figura 6.6: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

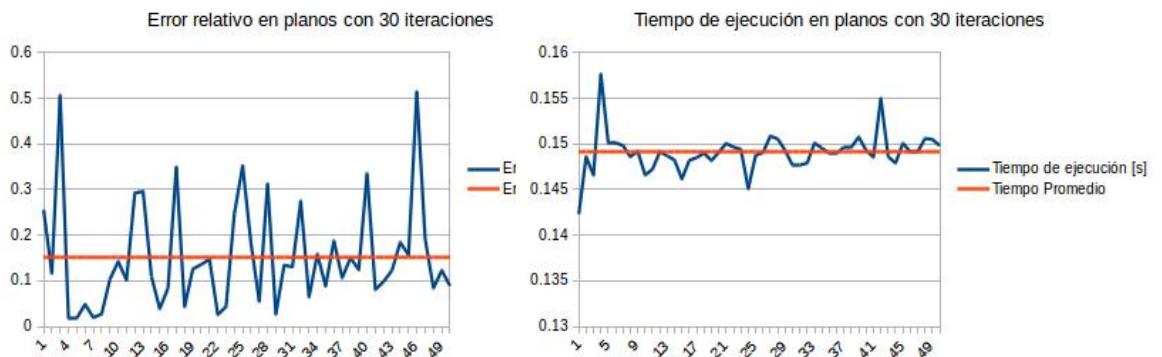


Figura 6.7: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

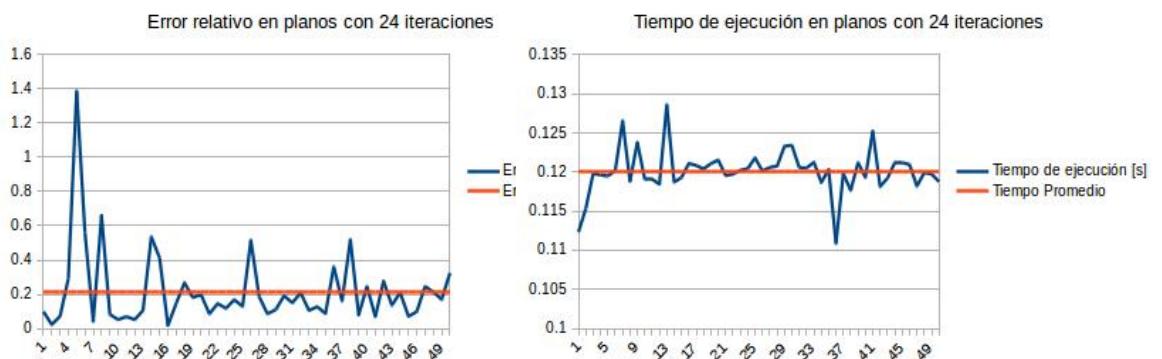


Figura 6.8: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

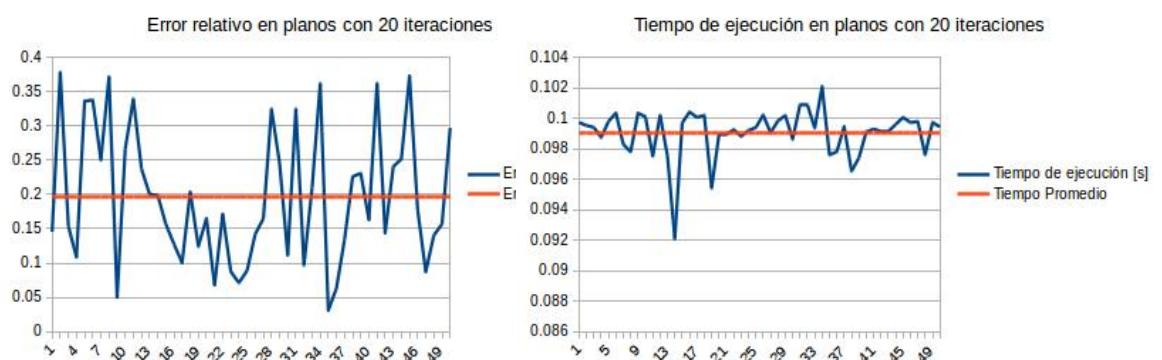


Figura 6.9: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

Una vez que se obtuvo la información parcial de cada uno de estos eventos se realizó una tabla extra que agrupa la información del tiempo de ejecución promedio y el error relativo promedio contra el número de iteraciones del algoritmo. Se obtuvo la siguiente gráfica.

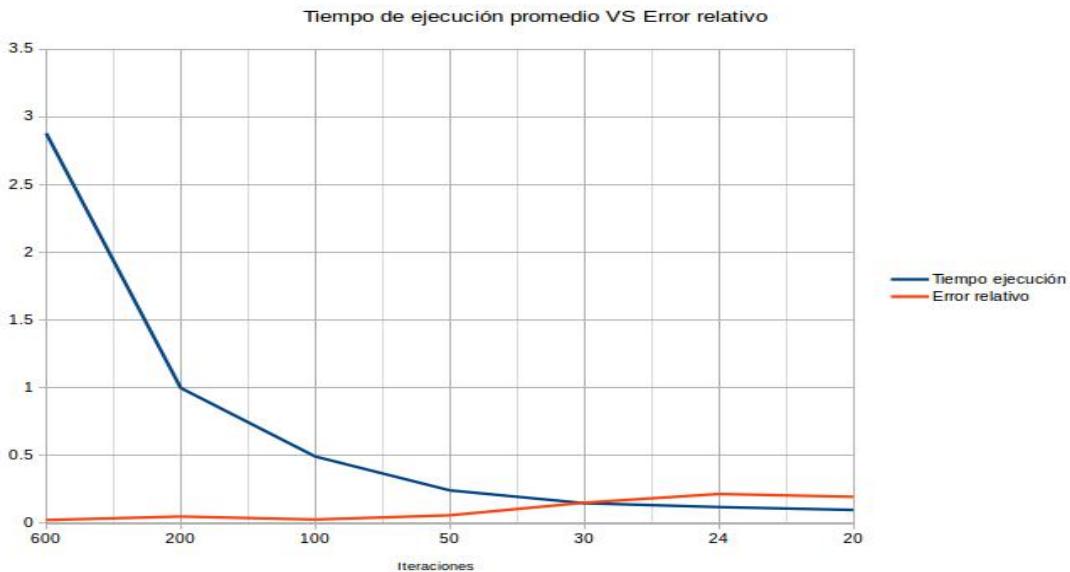


Figura 6.10: Segmentación de un objeto y sus correspondientes ejes principales (rojo).

Como podemos observar en la tabla 4.10 el error relativo presenta un incremento conforme se disminuye el número de iteraciones en el algoritmo. El tiempo de ejecución por su parte muestra un decremento conforme el número de iteraciones disminuye. Dado este comportamiento de ambos parámetros resulta difícil encontrar un punto de equilibrio entre estas dos unidades de medición. Sin embargo observamos que tanto el error relativo como el tiempo de ejecución llegan a un valor estable, a partir del cual no aumentan o disminuyen significativamente. La gráfica 4.10 nos sugiere que este punto está en un número de iteraciones entre 30 y 24.

Por otro lado, se procedió a caracterizar la extracción de objetos y el cálculo de su centroide. Para ello se realizaron un total de 5 eventos con objetos diferentes, 1 prisma octagonal y 4 objetos comunes con formas regulares: prismas rectangulares y cilindros. En cada uno de estos eventos se calculó el centroide del objeto 130 veces, se tomaron medida de tendencia central (valor esperado del centroide) y medidas de dispersión (varianza y

desviación estándar). A continuación se presentan los resultados obtenidos de las mediciones realizadas.

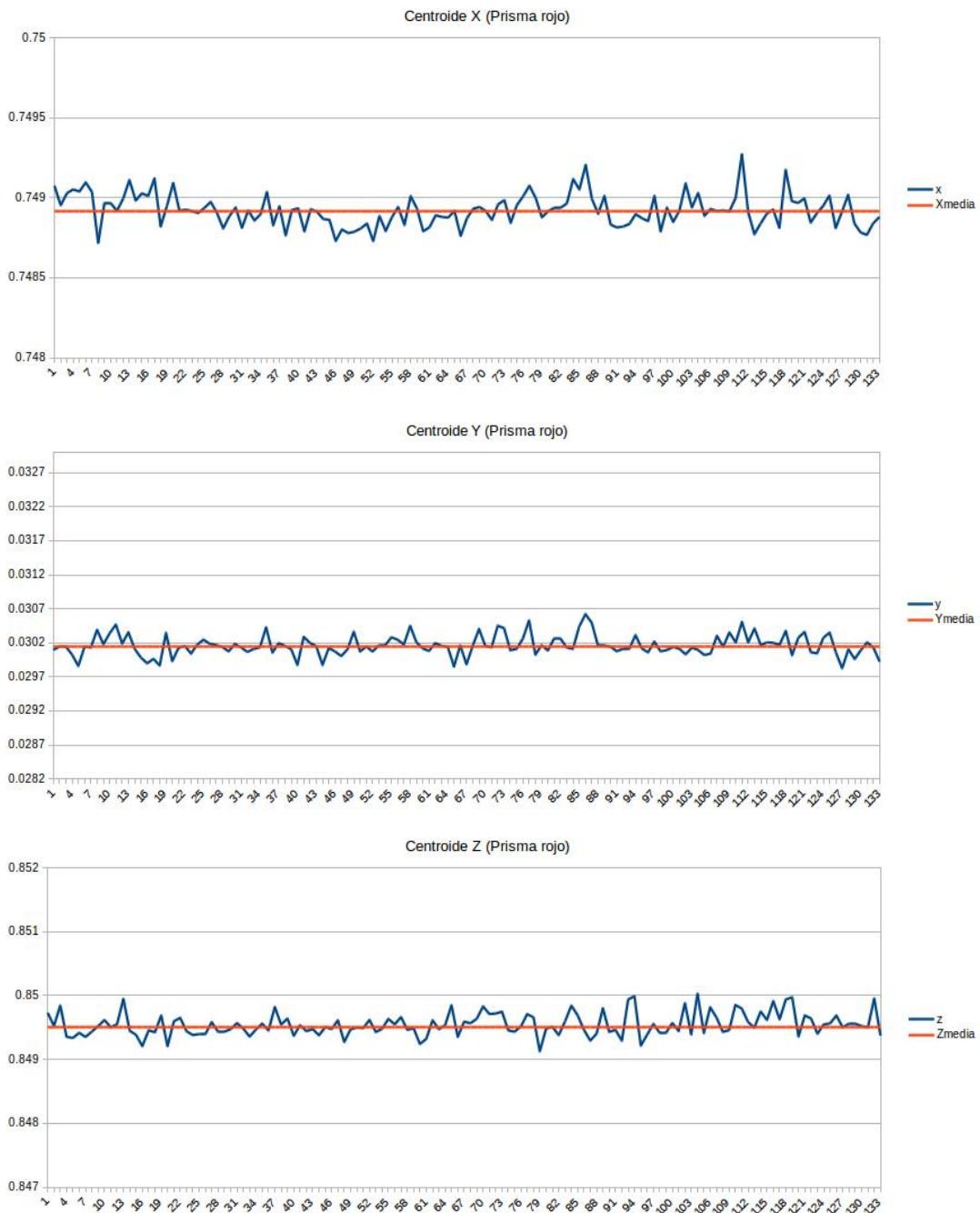


Figura 6.11: Gráficas de coordenadas en centroide de una barra de chocolate.

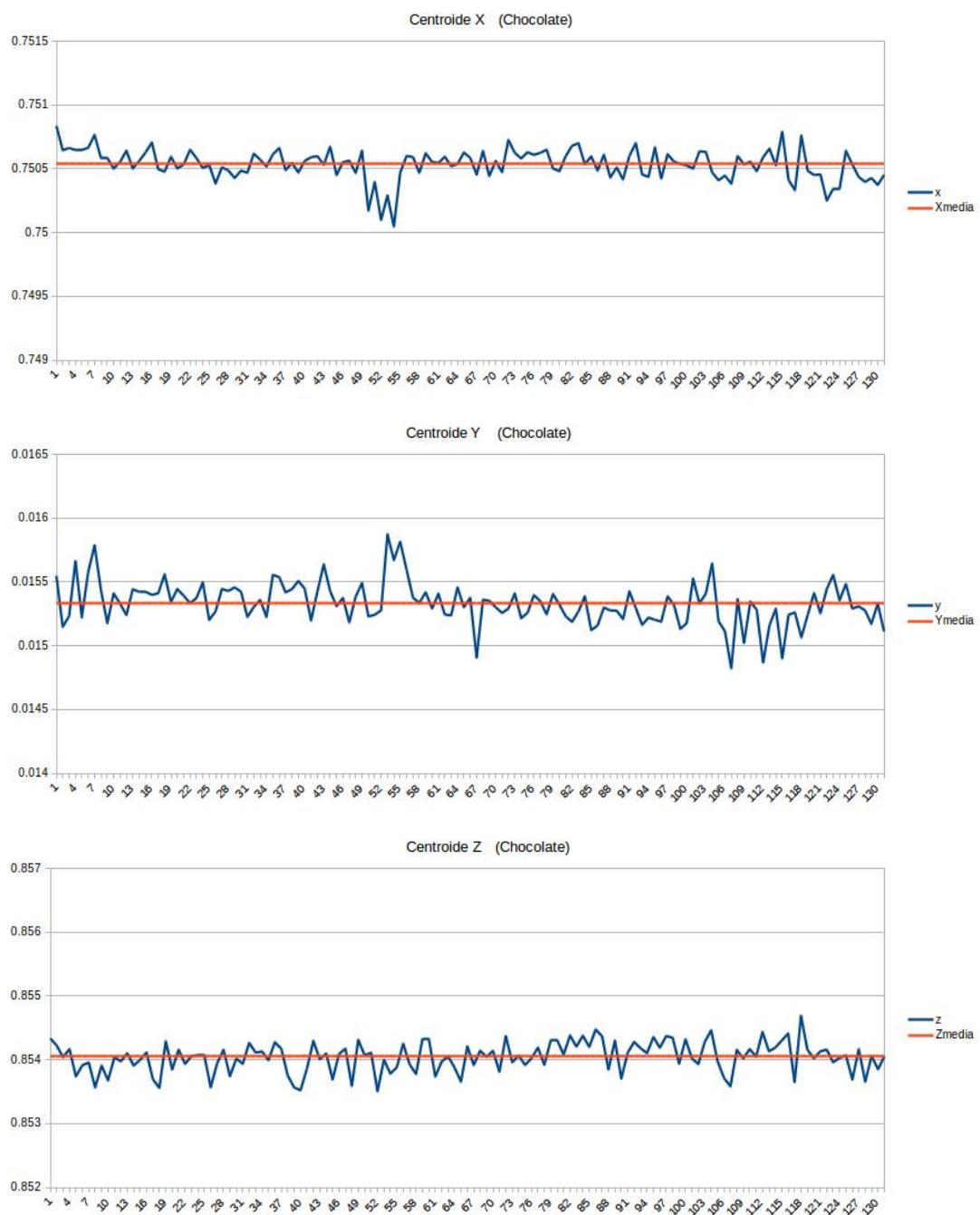


Figura 6.12: Gráficas de coordenadas en centroide de una barra de chocolate.

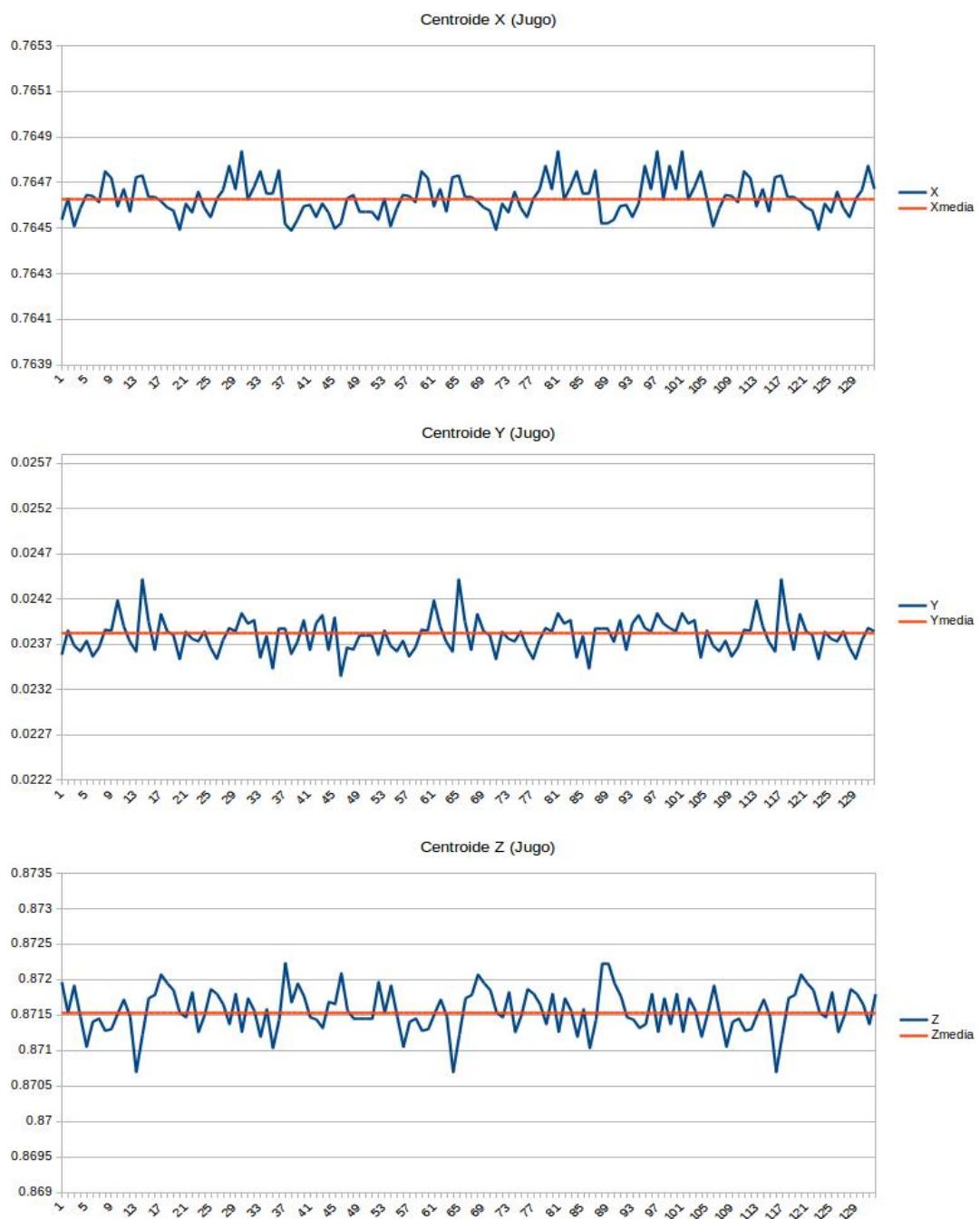


Figura 6.13: Gráficas de coordenadas en centroide de una barra de chocolate.

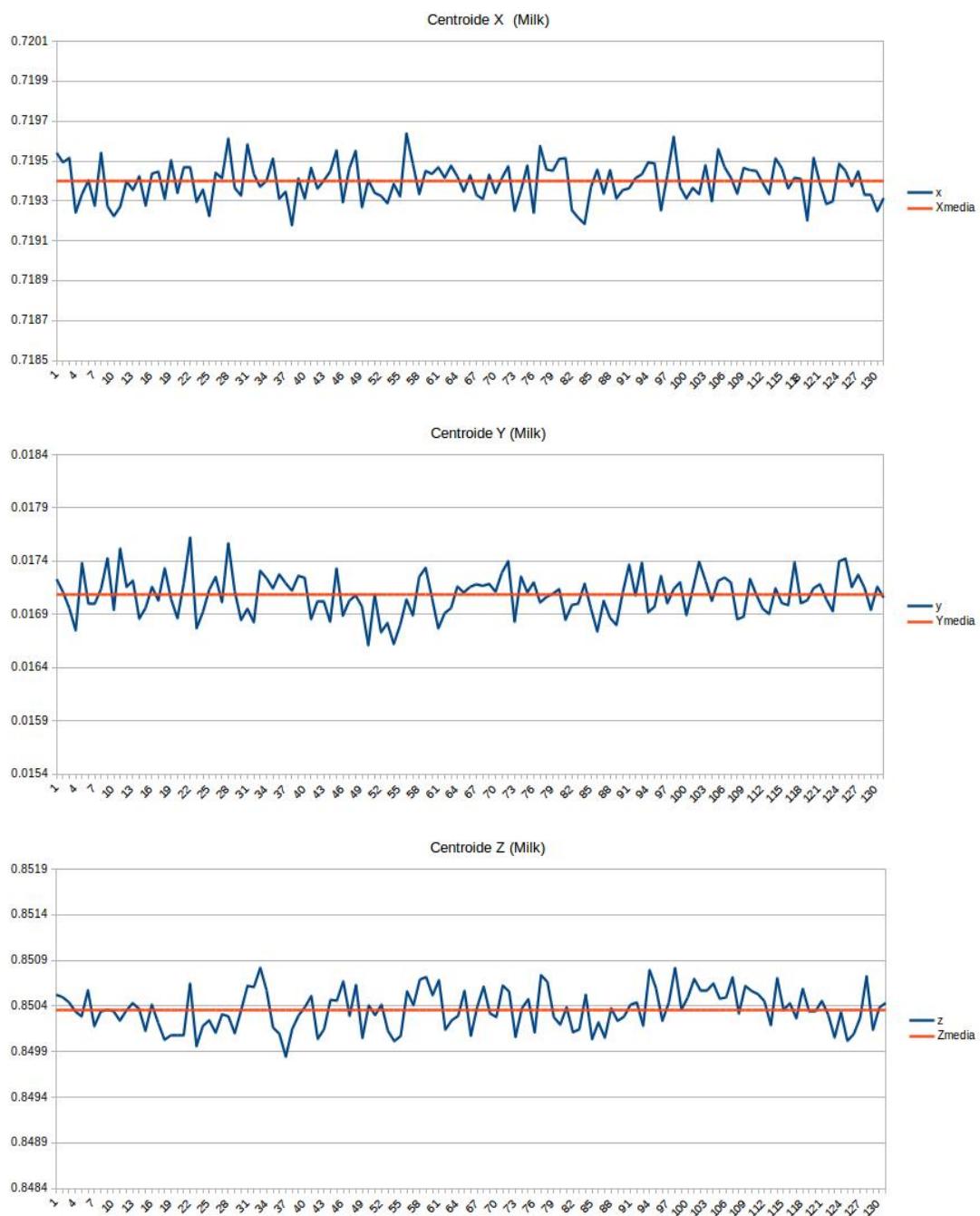


Figura 6.14: Gráficas de coordenadas en centroide de una barra de chocolate.

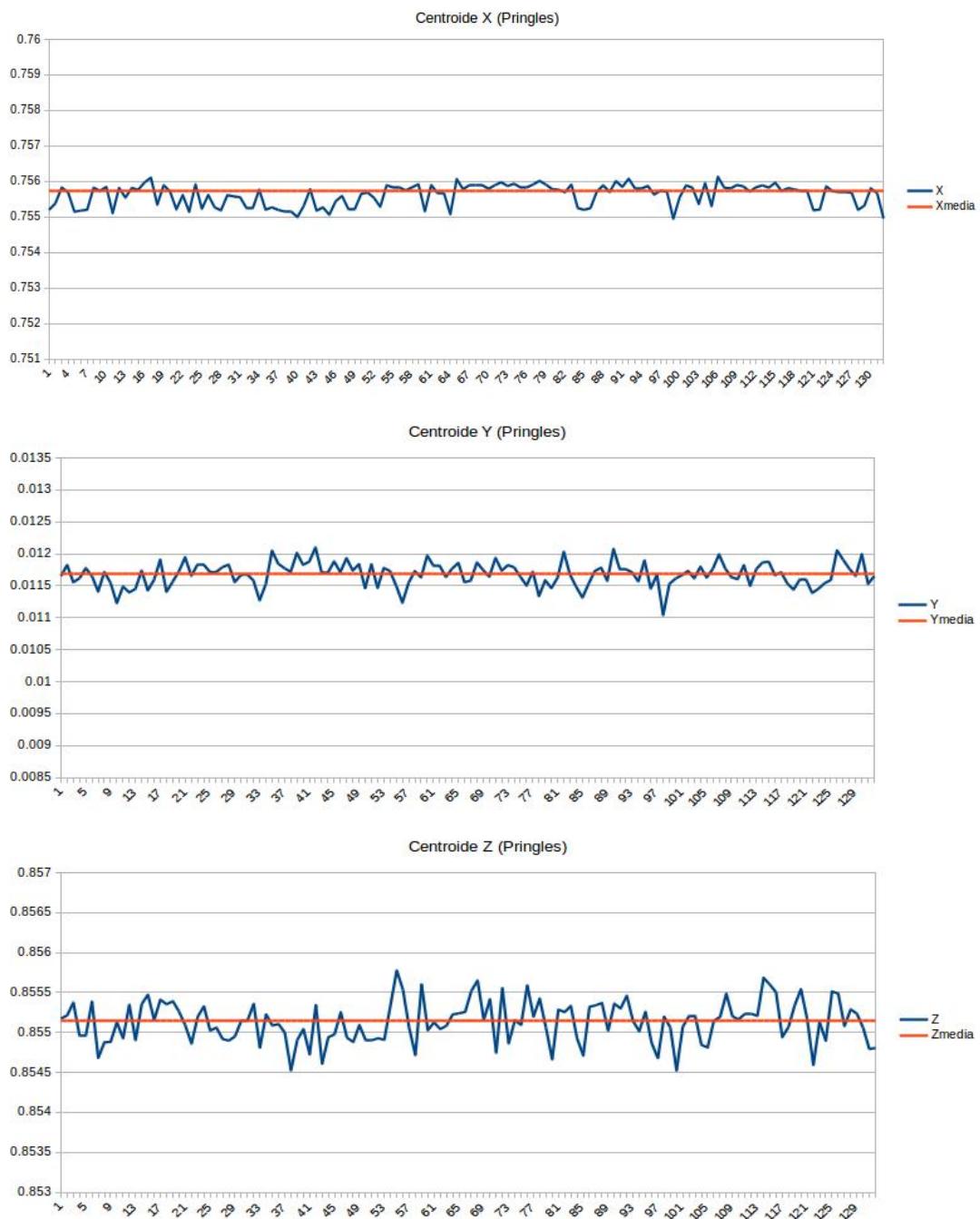


Figura 6.15: Gráficas de coordenadas en centroide de una barra de chocolate.

La desviación estándar puede ser interpretada como una medida de incertidumbre. La desviación estándar de un grupo repetido de medidas nos da

la precisión de éstas. Cuando se va a determinar si un grupo de medidas está de acuerdo con el modelo teórico, la desviación estándar de esas medidas es de vital importancia: si la media de las medidas está demasiado alejada de la predicción (con la distancia medida en desviaciones estándar), entonces consideramos que las medidas contradicen la teoría. Esto es coherente, ya que las mediciones caen fuera del rango de valores en el cual sería razonable esperar que ocurrieran si el modelo teórico fuera correcto. La desviación estándar es uno de tres parámetros de ubicación central; muestra la agrupación de los datos alrededor de un valor central (la media o promedio).

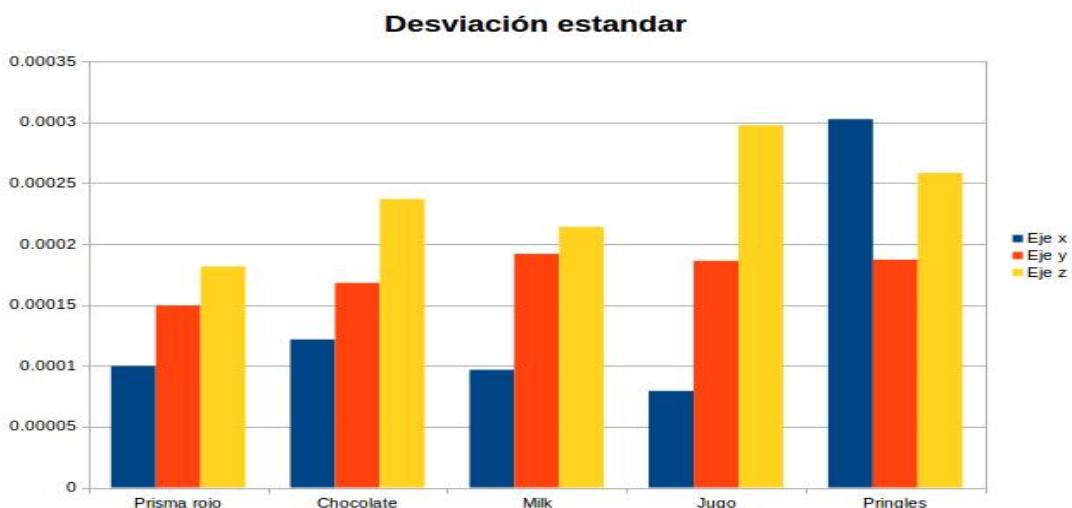


Figura 6.16: Gráficas de coordenadas en centroide de una barra de chocolate.

Partiendo de la información obtenida de los resultados podemos concluir que un algoritmo RANSAC con un umbral de 0.5 [cm] y un número de iteraciones entre 24 y 30 nos ayudará a encontrar la ecuación de un plano así como una lista de puntos pertenecientes al mismo. Esta primera parte del desarrollo permitió desarrollar un algoritmo que no presentara un tiempo excesivo de ejecución de manera innecesaria.

Una vez que se estuvo seguros que el algoritmo RANSAC funcionaba de manera aceptable se continuó a probar la extracción de objetos sobre el plano.

Para ello se tomaron 5 muestras de objetos diferentes. Se calculó la media de las coordenadas x , y , z y la desviación estándar de las mismas. En la Figura 4.15 podemos observar que el objeto que presentó una desviación

estándar de mayor magnitud es el objeto *pringles*, y de los tres ejes presenta una mayor desviación estándar en el eje "z". El orden de magnitud de la mayor desviación estándar es de 0.3 milésimas. Entonces, por el concepto de la desviación estándar podemos decir que el cálculo de la media de los centroides del objetos tiene un alto grado de cercanía a la predicción (media) del mismo. Por tanto el cálculo de la media en cualquier instante es confiable.

Bibliografía

- [1] W. Khalil and E. Dombre, *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.
- [2] “Sitio electronico oficial de la robocup at home.” <http://www.robocupathome.org/>. Consultado: 8-2017.
- [3] “Sitio electronico de la federación mexicana de robótica.” <https://www.femexrobotica.org/tmr2016/rctm-at-home>. Consultado: 8-2017.
- [4] J. M. Evans Jr, C. F. Weiman, and S. J. King, “Visual navigation and obstacle avoidance structured light system,” 4 1990. US Patent 4,954,962.
- [5] H. Endres, W. Feiten, and G. Lawitzky, “Field test of a navigation system: Autonomous cleaning in supermarkets,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, pp. 1779–1781, IEEE, 1998.
- [6] “Pagina principal de Ros.” <http://www.ros.org/>. Consultado: 10-2016.
- [7] “Robot Carnegie Mellon herramientas de navegación.” <http://carmen.sourceforge.net/>. Consultado: 10-2016.
- [8] “Página principal universidad Carnegie Mellon.” <http://www.cmu.edu/>. Consultado: 1-2017.
- [9] S. Balakirsky, C. Scrapper, S. Carpin, and M. Lewis, “Usarsim: providing a framework for multirobot performance evaluation,” in *Proceedings of PerMIS*, vol. 2006, 2006.
- [10] “Simulador gráfico rViz.” <http://wiki.ros.org/rviz>. Consultado: 11-2016.

- [11] “Simulador gráfico Gazebo.” <http://gazebosim.org/>. Consultado: 11-2016.
- [12] “Biblioteca abierta visión computacional.” <http://sourceforge.net/projects/opencv/>. Consultado: 10-2016.
- [13] G. R. Bradski and V. Pisarevsky, “Intel’s computer vision library: applications in calibration, stereo segmentation, tracking, gesture, face and object recognition,” in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, pp. 796–797, IEEE, 2000.
- [14] T. Wisspeintner and W. Nowak, *VolksBot-A Construction Kit for Multi-purpose Robot Prototyping*. INTECH Open Access Publisher, 2007.
- [15] “ActivRobots.” <http://www.activrobots.com>. Consultado: 10-2016.
- [16] “Robot de plataforma Standar Pepper.” <https://www.softbankrobotics.com/en/cool-robots/pepper>. Consultado: 11-2016.
- [17] “Robot de plataforma Standar Asimo.” <http://asimo.honda.com/>. Consultado: 11-2016.
- [18] A. O. Baturone, *Robótica: manipuladores y robots móviles*. Marcombo, 2005.
- [19] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. John Wiley & Sons, 2008.
- [20] J.-J. Hernandez-Lopez, A.-L. Quintanilla-Olvera, J.-L. López-Ramírez, F.-J. Rangel-Butanda, M.-A. Ibarra-Manzano, and D.-L. Almanza-Ojeda, “Detecting objects using color and depth segmentation with kinect sensor,” *Procedia Technology*, vol. 3, pp. 196–204, 2012.
- [21] Z. Zhang, “Microsoft kinect sensor and its effect,” *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.
- [22] “Sitio electronico oficial robotis.” <http://www.robotis.us/dynamixel/>. Consultado: 8-2017.